

3099

Troubleshooting and Repairing Your **COMMODORE**TM **128**



ART MARGOLIS

Troubleshooting and Repairing Your
COMMODORE™
128

Troubleshooting and Repairing Your
COMMODORETM
128

ART MARGOLIS

TAB **TAB BOOKS Inc.**
Blue Ridge Summit, PA

FIRST EDITION
FIRST PRINTING

Copyright © 1989 by TAB BOOKS Inc.
Printed in the United States of America

Reproduction or publication of the content in any manner, without express permission of the publisher, is prohibited. No liability is assumed with respect to the use of the information herein.

Library of Congress Cataloging in Publication Data

Margolis, Art.
Troubleshooting and Repairing Your Commodore 128 / by Art
Margolis.
p. cm.
Includes index.

ISBN 0-8306-9099-9 ISBN 0-8306-9399-8 (pbk.)

1. Commodore 128 (Computer) 2. Commodore 128 (Computer)-
-Maintenance and repair. I. Title. II. Title: Trouble shooting
and repairing your Commodore 128.

QA76.8.C645M37 1988
621.391'6—dc19

88-20129
CIP

TAB BOOKS Inc. offers software for
sale. For information and a catalog,
please contact TAB Software Department,
Blue Ridge Summit, PA 17294-0850.

Questions regarding the content of this book
should be addressed to:

Reader Inquiry Branch
TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0214

Contents

	Acknowledgments	viii
	Test Point Charts	ix
	Introduction	x
1	C128 Briefing Session The Symptom Devices—Diagnostic Programming—Printboard Landmarks—An Overview of Trouble Analysis	1
2	Disassembly Getting the Hood Up—Freeing the Printboard—The Power Supply Box—Visual Repairs—Cleaning Static Electricity	20
3	Chip Location Guide Specific Information—Chip Survey	39
4	Chip Changing Techniques The Rugged Chips—Tristating TTLs—The Sensitive Chips—The DIP Package—Socketed Chips—Soldered-in Chips	49
5	The LSI Chips The Microprocessors—The 6526 Complex Interface Adapters—The VIC and Video Controller Chips—8581 Sound Interface Device—The PLA and MMU	67

6	Dynamic Random Access Memory	86
	Static RAM—Dynamic RAM	
7	Read Only Memory	104
	Block Diagram—The Rest of the ROMs	
8	Other Integrated Circuits	118
	The 7406 Hex Inverters—The 7407 Hex Buffer/Drivers—The 74LS08 Quad 2-Input AND Gates—The 74LS00 and 74LS03 NAND Gates—The 74F32 and 74LS32 Quad 2-Input OR Gates—The 74LS74 Dual D Flip-Flop—The 74LS138 1-of-8 Decoder—The 74LS257 Multiplexers—The 74LS373 Octal Latches—The 74LS244 Octal Drivers—The Transceiver 74F245—The Schmitt Trigger 74LS14—The 4066 Quad Bilateral Switch—The 556 Dual Timer—Other Chip-Like Components	
9	The System Block Diagram	150
	Block Diagram—The 8502 and Dynamic RAM—The MPU and ROM—MPA and CIA Interface—The MPU and VIC—The MPU and SID—The MPU and the Video Controller	
10	Servicing the Logic Gates	167
	Decimal and Binary—Hexadecimal—Peek and Poke—Gates—Gate Testing Techniques	
11	Servicing Digital Registers	188
	Flip-Flops—Computing Registers	
12	8502 Microprocessor	207
	Address and Data Bus Lines—Stack—Arithmetic Logic Unit—Accumulator—Instruction Set—Index Registers—Flag Register—The 8502 Signals—Testing	
13	Z80 Coprocessor	231
	Z80 Block Diagram—Z80 Schematic Diagram—Z80 Instruction Set—Z80 Connections—The Computer Won't Start	
14	Programmed Logic Array	240
	PLA Internals—PLA Pinout	
15	Memory Management Unit	249
	Bank Contents—MMU Registers—Mode Configuration Register—Ram Configuration Register—The Page Pointers—Version Register—The MMU Pinout	
16	All the Memory Maps	266
	The C128 Maps—The C64 Maps—The CP/M Map—PEEK and POKE—MEMORY and FILL	
17	The Clocks	280
	Sine Wave to Square Wave—8502 Timing Control—Other Timing Signals—The Z80 Timing—Testing the Clock	
18	Address, Data and Control Buses	298
	All Those Address Buses—The Data Bus—The 80-Column Display Bus—The Control Lines—Testing the Bus Lines	
19	Complex Interface Adapters	321
	Addressing and Controlling—Internal Data Transfer—Timing—I/O Ports—•PC and •FLAG Pins—Timers—Real Time Clock—Interrupt Control Register—Serial Data Register—Operation—CIA1—CIA2—Testing	

20	8564 Video Interface Chip	337
	VIC Graphics—VIC Pin-by-Pin Operation—Sprites—Other VIC Features—Light Pen—Video Output—Testing	
21	80-Column 8563 Video Controller	354
	The 8563 RAM—Pin-by-Pin Checkout	
22	Sound Interface Device	364
	Pinout—Special Inputs—Operation—Registers—Timing—Testing	
23	Inputs and Outputs	379
	Checking Out I/O Troubles—RF Modulator	
24	The Power Supply	398
	First Step—Point-by-Point Checkout	
	Appendix	406
	Index	431

Acknowledgments

I'd like to thank my wife Lea for running interference and fielding interruptions, as usual, while I wrote the book. I would also like to thank my favorite photographer, my son-in-law, Michael Gorzeck, for taking the step-by-step disassembly photos

in Chapter 2. I hope your computing is uninterrupted, but should your C128 start acting up, perhaps this book will be directly responsible for getting it up and working once again.

Test Point Charts

U Number	Generic Number	Given Name	Figure	Table
U1	6526	Complex Interface Adapter	19-9	
U2	4066B	Quad Bilateral Switch	8-26	8-10
U3	74LS138	1-of-8 Decoder	8-16	
U4	6526	Complex Interface Adapter	19-10	
U5	6581	Sound Interface Chip	22-2	
U6	8502	Microprocessor	12-20	
U7	8722	Memory Management Unit	15-6	
U8	74LS08	Quad 2-Input AND Gate	8-6	8-3
U9	74F32	Quad 2-Input OR Gate	8-13	8-5
U10	Z80	Microprocessor	13-4	
U11	8721	Programmed Logic Array	14-6	
U12	74LS373	Octal 3-State D Latch	8-21	8-8
U13	74LS244	Octal 3-State Driver	8-22	8-9
U14	74LS257	Quad 2-Input Multiplexer	8-19	
U15	74LS257	Quad 2-Input Multiplexer	8-19	
U16	74LS14	Hex Schmitt Trigger	8-24	
U17	74LS373	Octal 3-State D Latch	8-21	8-8
U18	390059-01	Character ROM	7-3	
U19	2016	Color RAM	6-7	
U20	4066B	Quad Bilateral Switch	8-26	8-10
U21	8564	Video Interface Chip	20-1	
U22	8563	Video Controller	21-3	
U23	4416	DRAM	21-2	
U24	74LS244	Octal 3-State Driver	8-22	8-9
U25	4416	DRAM	21-2	
U26	74LS257	Quad 2-Input Multiplexer	8-19	
U27	556	Timer	8-28	
U28	8701	Clock	17-16	
U29	7406	Hex Inverter Buffer	8-4	8-1
U30	7406	Hex Inverter Buffer	8-4	8-1
U31	74LS00	Quad 2-Input NAND Gate	8-9	8-4
U32	251913-01	Read Only Memory	7-6	
U33	318018-02	Read Only Memory	7-6	
U34	318019-02	Read Only Memory	7-6	
U35	318020-03	Read Only Memory	7-6	
U36	Empty Socket for	Function ROM	—	
U37	7406	Hex Inverter Buffer	8-4	8-1
U38-U53	4164	DRAMs	6-1	
U54	74LS32	Quad 2-Input OR Gate	8-13	8-5
U55	74F245	Transceiver	8-23	
U56	74LS74	Dual D Flip-Flop	8-14	
U57	7407	Hex Buffer	8-5	8-2
U58	74LS03	Quad 2-Input NAND Gate	8-9	8-4
U59	7812	12 Volt Regulator	24-3	
U60	7407	Hex Buffer	8-5	8-2
U61	74LS08	Quad 2-Input AND Gate	8-6	8-3
U62	74LS244	Octal 3-State Driver	8-22	8-9
U63	7406	Hex Inverter Buffer	8-4	8-1

Introduction

After I reassemble a computer like a C64 or C128 following a repair job, I usually type in some little test program and run it. One of my quick favorites is this two line tester:

```
10 ?RND(1);  
20 GOTO10
```

The test lines cause the screen to fill up with a stream of scrolling decimal numbers. This tells me that most of the chips in the machine are okay because they are participating in the action. It is a good, fast checkout exercise.

One time, after taking the step-by-step disassembly photos for Chapter 2 and reassembling my C128, I routinely typed in the test lines. The line number 10 appeared fine, but when I hit the ? key to command a PRINT, nothing happened. I tried the ;. It didn't work either.

Instead of using the ?, I typed in PRINT and left off the semicolon. Then I ran the lines. Instead of filling the entire screen with decimal numbers, a

row of decimal numbers started scrolling up the left side of the screen.

I breathed a sigh of relief. The C128 itself was okay. It looked like the trouble was strictly related to the keyboard. I took the top off the machine again and spotted the trouble. The keyboard port plug was not seated properly. I had evidently pulled the plug out a fraction during the final moves of the reassembly. I seated it firmly and put the screws back in. Sure enough, all was well again.

Easy-to-fix trouble is typical of the majority of computer problems. If you develop a simple problem and take it into a repair shop, you'll be presented with a bill—after the shop gets to your C128, takes it apart, diagnoses the circuit area, pinpoints the cause, and remedies it. In addition, days and even weeks could elapse without the use of your computer.

To avoid a lot of expense and hassle, all you have to do is learn how your C128 works, from the viewpoint of the hardware. Even though you might be a top programmer or an expert computer opera-

tor, the troubleshooting and repairing of your C128 can only be accomplished by approaching the machine from a hardware point of view. It will be easier to gain the repairer's viewpoint if you are a programmer or operator, but the hardware approach is different from the software approach, and requires a different dimension of thought.

A car as an extension of your legs is analogous to a computer as an extension of your brains, and driving a car is analogous to programming a computer. The average auto driver knows little about how the car's engine burns gasoline. In the same way, the programmer has only a smattering of what the computer's circuits are doing as it consumes electricity. When the auto breaks down, the driver takes it to the mechanic to be fixed. When the C128 breaks down, the programmer looks for a computer troubleshooter.

In one respect, this book, besides covering routine, easy repairs, is the manual the computer technician uses to make the necessary tests, no matter how difficult, to get your C128 "up" and working again. However, this book is much more than that. Besides containing the specific information on voltages, logic states and parts, this book is also a training course to give Commodore 128 owners the information that, added to programming skills, will give you the missing hardware point of view that will enable you to gain complete mastery over your computer.

In order to be able to figure out and repair a circuit failure in your C128, first of all you must be able to picture in your mind how the circuit is working. Tried and true electronic service pathways that have been developed over the years are useful. By following these methods, and using your natural puzzle-solving abilities, you will be able to pinpoint defective components or connections. Once the trouble is located—the most trying part of a repair job—then the rest of the chore is either replacing the part or repairing the connection.

The starting point of any repair is to carefully analyze the symptoms of trouble. One symptom is the computer not being able to display all the keys on the keyboard. A number of others exist. In Chap-

ter 1, all the common symptoms are discussed. An analysis of each symptom points to a circuit area that could possibly contain the source of the trouble. In each circuit, there are primary suspects that should be examined first. However, in order to get to the circuits, you must be able to take the C128 apart. It is not difficult, and Chapter 2 goes into the details; it's all about knowing where the screws are.

You must use the right tools for the disassembly. Dismantling and reassembling must be performed correctly, and with care, to avoid mishaps. You are working with tiny items, and each move must be made slowly and then it must be checked. A wrong move could cause "induced" troubles that will complicate things.

Chapter 3 contains the most used piece of service information: the chip location guide—the main printboard layout of the 63 chips in the computer, plus other prominent landmarks. The guide lets you relate all the information you learn to the location of any chip you might want to examine on the board, and is used over and over again on every repair job. Its use results in many repairs without any other service information.

Chapter 4 provides the mechanical and electronic techniques that must be used if and when chips have to be replaced. The tools and thinking that go into the ticklish job of changing an integrated circuit chip is reviewed. Changing a chip is not like changing a vacuum tube, and is similar to, but much more exacting than, changing a transistor. These four chapters will teach you how to quickly repair at least 50 percent of C128 failures.

Chapters 5 through 8 introduce you, with more intimacy than previous chapters did, to the chips in your machine. Starting in Chapter 6, you'll find the first of the test point charts that make it a simple matter to quickly check out every chip in the computer. Each chart is the top view of a chip showing the exact pinout. The name of each pin is given and, where practical, a sketch of the chip's insides is also shown. Arrows show the direction of the signal flow. Attached to each arrow is the actual reading you should receive if you probe the test point with a logic probe or *vom* (voltohm-milliammeter).

The readings were made on a computer that was first turned on in C128 mode, 40-column, with the READY sign on the screen and with the cursor blinking. When you read your chips, the logic state of each test point should match up with the charts. If all the pins on the chip read correctly, then that chip is deemed okay. Should one or more readings on a chip be incorrect, then this is a symptom of trouble and bears further investigation.

Actually, what you are doing is checking the input and output status of the chips. The inputs and outputs are clearly shown by the arrows. If all signals are being input properly but are not outputting correctly, chances are that the chip's internal circuits are not passing the signals. That would indicate a bad chip. On the other hand, if an input signal is incorrect, odds are that the chip is okay but that the circuit feeding the chip is not getting a signal to the pin. That circuit or bus line could have a short or an open condition. The test point charts are your entree to quickly getting repairs underway and locating troubles.

Chapters 5 through 24 comprise a servicing manual and a technical reference manual written on a technician's level. There are discussions of each chip with considerable detail, and even some timing diagrams for the chips that require them. You will gain mastery over your C128 by gradually absorbing the material. You might find that your programming skills will also improve as you gradually realize what is actually happening to the 1's and 0's as they flash around the digital circuits.

This book ends up with a master schematic of the C128. The schematic is needed during a difficult repair after the circuit area containing the trouble has been located and the wiring details are needed to pinpoint the prime suspects. The schematic contains all the part numbers. These same part numbers are found printed clearly on the print-

board. For example, a U6 is found printed on the board next to the 8502 MPU. The schematic reads U6 as the part number for the 8502. All the transistors, capacitors, etc. are identified in the same way. With the chip location guide, the test point charts, the theoretical discussions of the circuits, and the master schematic, you should be able to cover all the information required for the troubleshooting and repair of your computer.

Chapters 10 and 11 are a short course in logic gates and digital registers with specific reference to the gates and registers in the C128. The way the machine uses binary, hexadecimal and decimal is included too. This knowledge enables you to work out BASIC PEEK and POKE routines to signal trace circuits that the 8502 MPU is able to address. The C128 also has a machine language monitor that can also be used for signal tracing with its commands. The ability to convert from decimal to binary to hex and back should be one of the tools of your servicing repertoire.

To be able to make the easy repairs on your C128 really pays off. The easy repairs are at least half of the total repairs. While your C128 is still in warranty, there is no problem—Commodore is very accommodating. However, once the warranty period is over, the complication begins. If the machine fails, you must take it to a repair station. This often takes some weeks, and the machine can be returned with a healthy repair bill. If, on the other hand, you are able to do the troubleshooting and repair yourself, then the savings in time and money become quite worthwhile—besides the feeling of accomplishment you get when you fix your machine. If, during the life of your machine, you happen to get one repair from this book, the savings will more than pay for the price of the book. Then, of course, there is the knowledge you'll gain from the experience.

1. C128 Briefing Session

The unusual thing about troubleshooting a C128 in comparison to trying to fix most other computers is that the C128 case contains a form of Siamese triplets. Three complete computer systems are in the one case. The three computers are joined together at their inputs, outputs, *MPUs* (Micro-Processing Units), and memory. They also share many of the other circuits. They cannot be physically separated from each other.

The triplet feature of the C128, as might be imagined, can complicate and confuse the situation when trouble strikes. However, if you are properly briefed and are aware of the three-in-one computer feature, as shown in Fig. 1-1, then symptom analysis can be accomplished without difficulty, and that is what this book is all about. It covers the three computers in the C128 case from a troubleshooting point of view.

THE SYMPTOM DEVICES

The input and output devices of your computer are diagnostic tools. The most valuable of them is

the TV display. Most computer troubles show up on the TV display. In many cases, you can look at what is happening, or not happening, in the display and be directed to the circuits that are in trouble.

Other input or output devices that could also aid in interpreting symptoms are: the printer, the disk drive, the cassette, the keyboard, the modem, a cartridge, the joysticks, etc. When these devices "act up," you have to decide whether the computer is at fault or the device is bad. The decision is easy if you substitute a device that is known to be good for the one exhibiting the symptom. If the new one works properly, then the old one is defective. If the trouble persists, the finger of guilt points to the computer circuits. Table 1-1 is a trouble chart that indicates the computer circuit that could be the problem when peripherals stop operating, but are proven to be good. These input and output devices are excellent symptom indicators. However, as mentioned, the TV display reveals the most symptom information. The symptoms that displays show become classics, have names of their own, and contain valuable

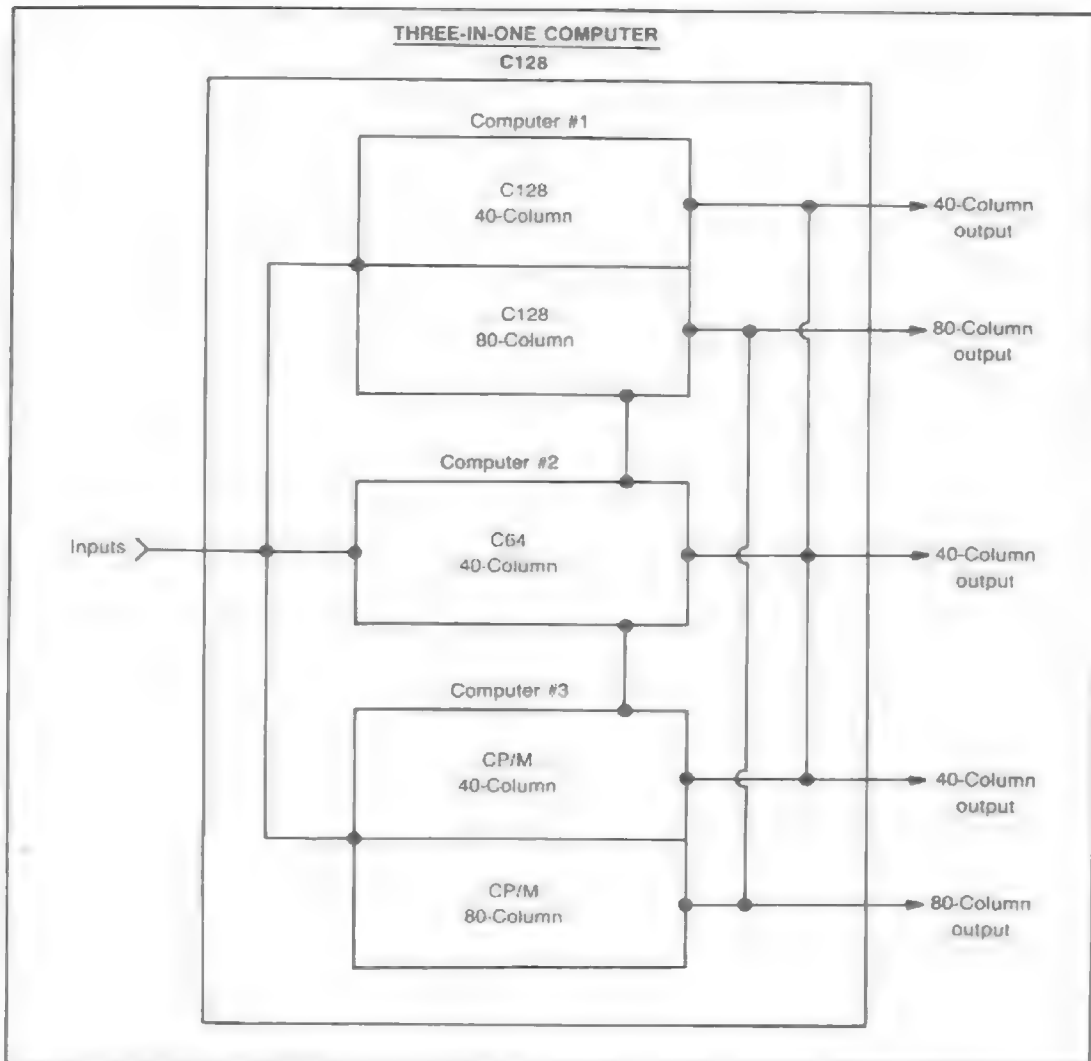


Fig. 1-1. Troubleshooting the C128 can become confusing because there are three separate computers joined together like Siamese triplets.

troubleshooting information when interpreted properly. Let's go through them one by one.

The Items in a Normal Display

Before you can spot trouble in a display you must know what the display should look like when

there isn't any trouble. Table 1-2 shows the five possible types of display devices you might be using. They are (1) your home TV, (2) a monitor that displays a composite color TV signal, (3) a monitor that needs an RGBI signal, (4) a monochrome monitor, (5) a direct monitor.

Table 1-1. When a peripheral won't work because of computer trouble, these chips are the prime suspects.

Symptom	Possible Defect in C128	Try
Disk won't work	CIA2	Replacing chip
Printer stops	CIA1/CIA2	Replacing chip
Keyboard won't work	CIA1/ or Keyboard	Replace chip or repair or replace keyboard
Modem not operating	CIA1/CIA2	Replacing chips
Loses audio	SID	Replacing chip
40-column display loses video	VIC/RF Modulator	Replace VIC or replace or repair RF Mod box
80-column display loses video	8563/74LS244/4416's	Replacing chips
Cartridges won't work	CIA2/MMU/PLA	Replacing chips
Control port won't work	CIA1	Replacing chip
Cassette won't work	8502/CIA1/7406,U30	Replace chips or transistor

Table 1-2. The C128 outputs five different signal types from its two video output circuits. One type of signal is for each of five different kinds of monitors.

Available Modes	Output Circuits	Display Device
C128 40-column	RF Modulator 8564 VIC	Home TV Composite Monitor Direct Monitor
C128 80-column	8563 Video Controller	RGBI Monitor Monochrome Monitor
C64 40-column	RF Modulator 8564 VIC	Home TV Composite Monitor Direct Monitor
CP/M 40-column	RF Modulator 8564 VIC	Home TV Composite Monitor Direct Monitor
CP/M 80-column	8563 Video Controller	RGBI Monitor Monochrome Monitor

Because most people use their home TV, I'll use it as the example. In general though, the discussion covers all five types. The differences are covered in detail in Chapters 20 and 21.

The C64 mode of the C128 is built to come on as shown in Fig. 1-2. Item number 1 in the picture is a light blue border. Item number 2 is a dark blue display block. Item number 3 is a sign-on message

ending with READY. Item number 4 is the blinking cursor. These signals emerge from the RF socket in the back of your C128, passing through the cable and entering the TV via the Computer/TV switch-box, into the VHF antenna terminals of the TV.

There is one more signal from the C128 that is not displayed. The above four signals are packaged up in a TV frequency set for either Channel 3

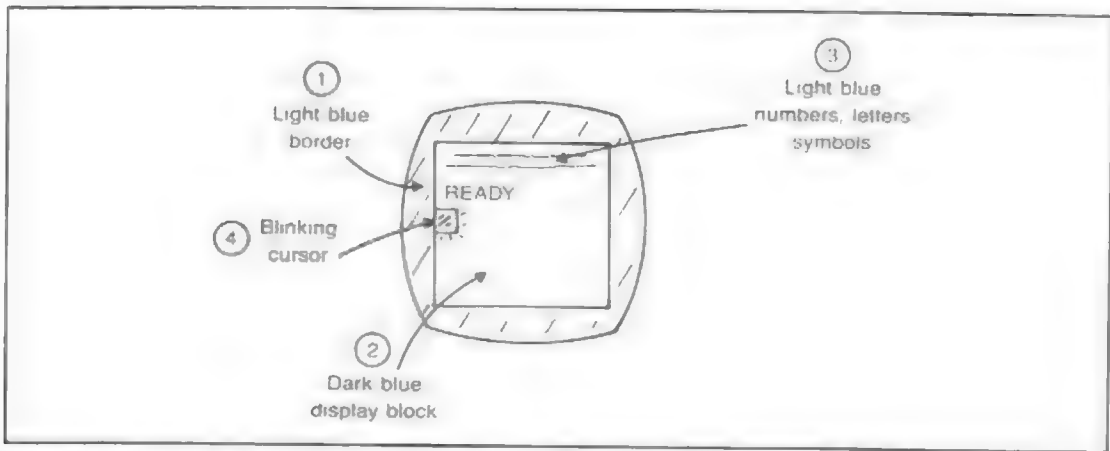


Fig. 1-2. In C64 mode the display normally comes on with a light blue border, a dark blue display block, light blue characters, a READY prompt and a blinking cursor.

or 4. The Items are modulated in these TV channel frequencies and are then demodulated in the TV so that they may be displayed on the screen. The modulation-demodulation process degrades the signals slightly. That is why the monitors are preferred over a home TV display. For monitors, the signal items are sent directly to the monitors through the composite video socket or the RGBI socket without having to go through the modulation-demodulation in the RF Modulator circuits. The home TV is quite satisfactory and handy.

When you turn on your C128, you should see with the 40/80 key in the up position, and in C128 mode, a black block, a green border, green lettering and a cursor. If anything is missing, then you are experiencing trouble. Incidentally, if the block and the border show, but the lettering and cursor are missing, then the first thing you should do is check the 40/80 key. It should be in the up position. If it is depressed, it could cause the lettering and cursor to be missing on your home TV screen.

When the 40/80 key is up, the signal is coming out of the RF socket and shows all four Items on the display. However, when the 40/80 key is depressed, the signal is switched around and the four Items appear out of the RGBI socket. The RF socket only puts out the display block and the border.

The Dead Computer

The most common problem with the C128 is a complete loss of power. When this happens, the TV display shows a blank screen. If the display is a home TV, then the picture will appear as if there isn't any antenna attached. Most TVs at that time will show a screenful of snow, or some snowy distant channel as shown in Fig. 1-3. When you flip the C128 off-on switch, nothing shows on the display. The red pilot light might or might not go on.

You must make the obvious service moves first. Check to be certain that the computer box is plugged properly into the wall socket on one side, and the computer on the other, as seen in Fig. 1-4. If the plugs are okay, then feel the casing of the power box. Is it slightly warm? If it is, then the box is prob-

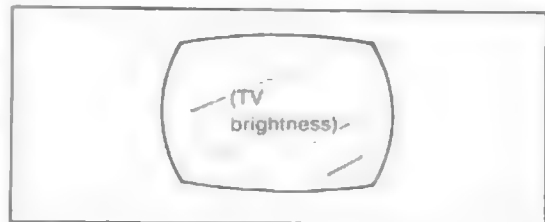


Fig. 1-3. When your home TV or monitor comes on as usual but your C128 won't display anything, your computer is playing dead.



Fig. 1-4. The first obvious step to take when the C128 is playing dead is to make sure that the power box is plugged into the wall socket and into the side of the computer.

ably okay. If it is stone cold, then it could be the troublemaker. Trying a new one will prove the point. If a new one fires up the computer, then the old one is bad. Should the new one not work either, then the old one is probably fine.

Once you complete these quick checks and the computer is still dead, your next step is to turn to Chapter 24. There you will find the next troubleshooting steps. You'll need to take some ac and dc voltage readings, and probably disassemble the power box and the C128.

Garbage Display

The next most common symptom is loosely known as "garbage." It is generally thought of as a screenful of meaningless numbers, letters, symbols, white spaces and black spaces as shown in Fig. 1-5. The trouble could happen during start up. The display fills up with garbage instead of the normal sign-on with the READY prompt. The garbage could contain a flashing cursor, or the cursor could also be missing. Other times the garbage could suddenly appear while you are computing. Your program crashes and the computer locks up.

The reason why garbage appears on the screen, instead of the normal sign-on picture, is because the microprocessor is out of control—it is running wild. Normally the processor conducts its duties under

the careful control of the C128's operating system in the ROM (Read-Only Memory) chips. When for some reason the operating system loses control, the processor simply goes mad, spewing addresses, data and control signals around the computing circuits without any rhyme or reason. The result is: the TV display fills up with nonsensical characters, numbers, symbols and spaces.

The garbage symptom does not pinpoint the trouble to a single circuit. The failure could happen in almost any of the digital circuit components or chips. Your approach therefore must be: first, locating the general circuit area of the trouble before you can zero in on a specific circuit; second, locating a component, connection or chip.

This procedure requires you to learn how the digital circuits are processing data from a hardware and voltage point of view, at the technician's level of understanding. Then you can intelligently make appropriate logic probe, voltage and scope readings to hunt down defects. Chapters 6 through 23 contain the information that will aid you in pinpointing the troublemaker.

Empty Display Block

The empty display block is the same symptom as what occurs when the 40/80 key is accidentally depressed. As shown in Fig. 1-6, the border and dis-

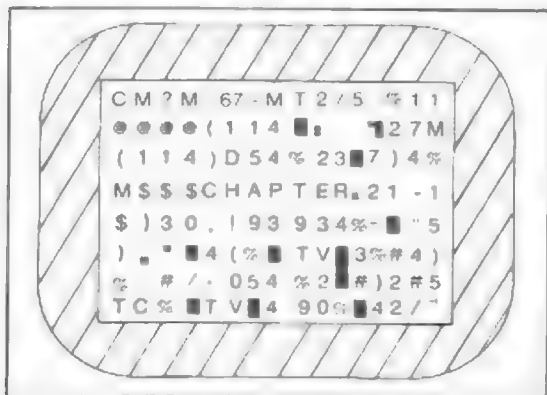


Fig. 1-5. When the display fills up with GARBAGE instead of the READY sign-on message and the blinking cursor, the trouble is hidden somewhere in the digital circuits of the computer.

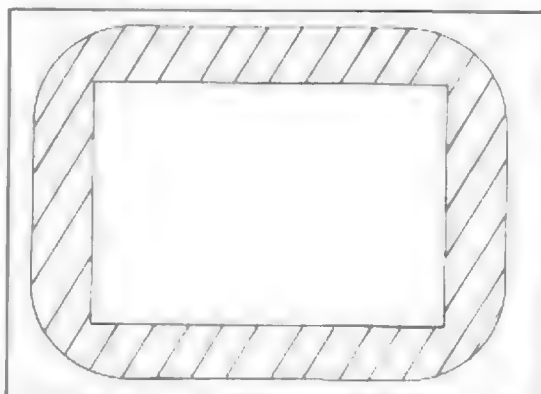


Fig. 1-6. A variation of garbage is a display block that is blank. The border and empty block can be seen but no characters appear. Striking the keyboard has no apparent effect.

play block are present but the block is completely empty. You can strike the keyboard but try as you may, nothing shows up—the block remains devoid of characters or symbols.

When the display block is empty and the 40/80 key is up, you have a circuit problem that results in another form of garbage. Here again the same circuits come under suspicion. The fault could be located anywhere in the digital circuits. It is a troubleshooter's job to take test readings and, from the results of the readings, deduce what circuit area is in trouble. Again, an understanding of Chapters 6 through 23 should clear up the maze of chips and printboard copper traces.

Chapters 6 through 23 contain troubleshooting techniques and theory of operations along with detailed test point charts showing voltage and logic states that are normally present on the test nodes when the READY sign appears. What you do is run voltage or logic probe tests, as described in Chapter 4. If you take some test readings and one or more of the test results do not match up with what is supposed to be at the connections, then you have found a clue that could lead to the fault.

No Color

The C128 is a color computer. Special circuits in the machine generate the colors that appear in the display. Should the C128 display a picture with good characters and symbols, except for the fact that the picture is missing coloring, then no color indicates trouble in the color circuits.

The color signal is originated in the clock circuits. Chapter 17 covers the clock. The colors are

output for the 40-column displays in the 8564 VIC chip. Chapter 20 explains the 8564 chip. The colors for the 80-column displays are output by the 8563 Video Controller chip. The discussion on the 8563 is in Chapter 21. When there is no color, turn to these chapters to find out how the color is generated and how it passes through the computer. Once you brief yourself on the color theory of operation, you can then figure out how the coloring is failing to appear and take the repair measures to restore color.

No Video, Sound Okay

This symptom is almost like the dead computer symptom except for one important difference: the sound from the computer is still emanating from the TV display. This means that the computer is fine, except for the video circuits. Because there are a number of different types of video outputs possible from the C128, you must make some simple isolation tests to determine which videos are missing and which ones are present. The possible video outputs you can obtain from the C128 are the following.

The RF output plug sends a composite TV signal contained in a TV Channel 3 or 4 envelope (see Fig. 1-7). A home TV can use this signal just as if it was coming from a TV station. This RF output has a 40-column display only.

The Video output plug is tricky. It puts out two different type 40-column signals. First of all, it outputs through pin 4 a composite TV signal without the TV Channel 3 or 4 modulation envelope (see Fig. 1-8). This signal can be displayed on a composite monitor TV. Secondly, the plug outputs another dual

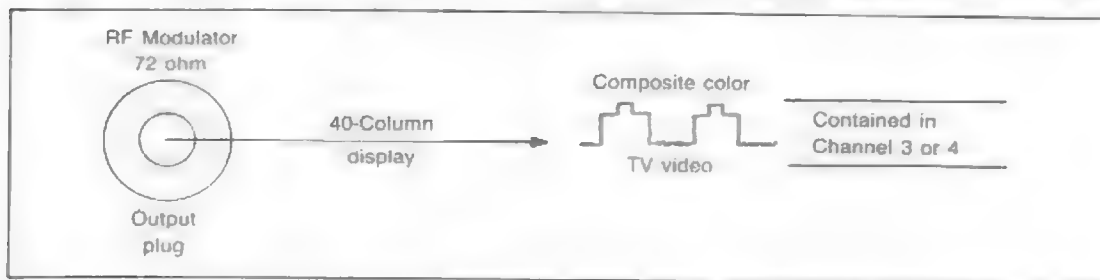


Fig. 1-7 The RF Modulator output plug sends a 40-column composite color TV signal on a Channel 3 or 4 frequency. It works fine on a home TV

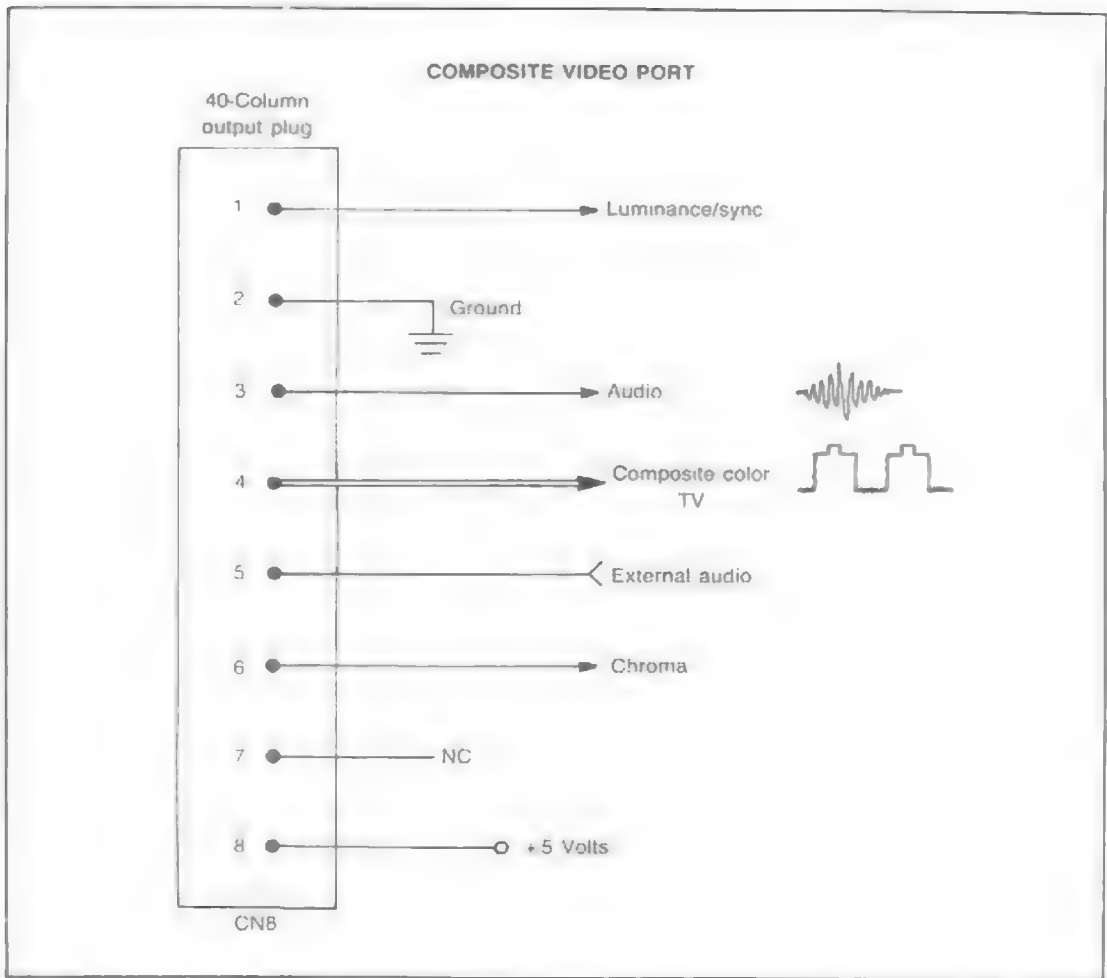


Fig 1-8 The Composite Video Port sends out a number of signals. At pin 4 is a 40-column composite color TV signal without Channel 3 or 4 modulation. At pins 3 and 5 are audio output and input signals. Pins 1 and 6 output separate 40-column LUM/SYNC and CHROMA.

signal. From pin 1 it emanates a Luminance/Sync signal. From pin 6 comes a Chroma signal. These two signals can be injected into a direct TV monitor to produce a display. There aren't too many of these monitors around except for the Commodore 1702 and perhaps a few others. However, the combination signals are available at the pins and can prove useful during the testing of video signals. Any ordinary TV scope will show them nicely.

The RGBI plug is also not straightforward. It also outputs different signals (see Fig. 1-9). It puts out a normal 80-column RGBI signal that can drive an RGBI Monitor. The RGBI signal exits through most of the nine pins of the RGBI plug. In addition, another separate 80-column signal is output from pin 7 of the RGBI plug. It is an 80-column Monochrome signal that can be used on an ordinary composite TV monitor. It is similar to the 40-column signal com-

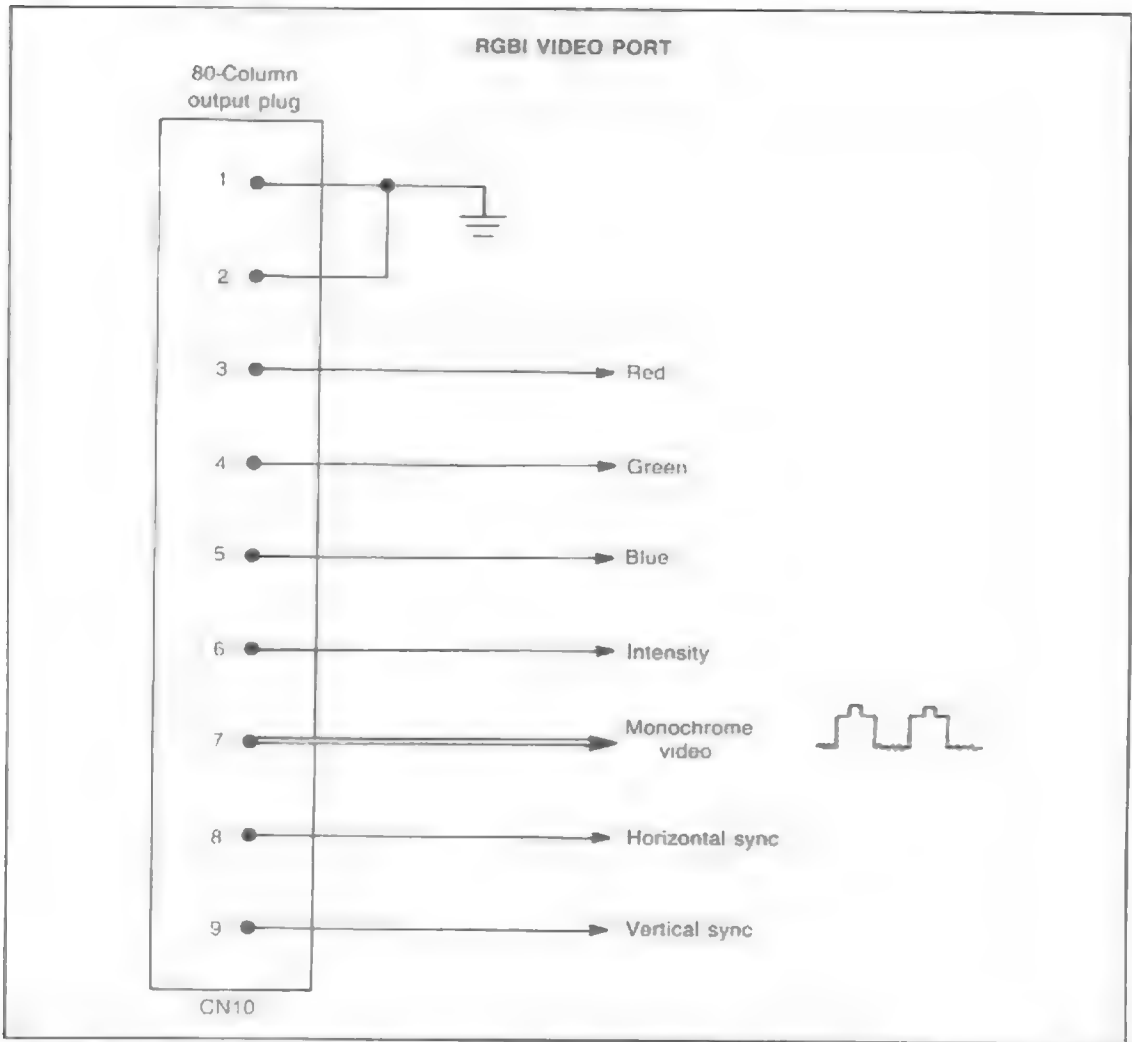


Fig 1-9 The RGBI Port outputs two separate signal-types. From pin 7 emanates an 80-column Monochrome Video. All the rest of the signals combined produce an 80-column RGBI signal with correct horizontal and vertical sync.

ing out of pin 4 of the composite video plug—only this pin 7 output from the RGBI plug has 80 columns. The columns are different because the 40-column output comes from the VIC 8564 chip and the 80-column output comes from the 8563 video controller chip.

Table 1-2 shows the different outputs. The isolation test techniques are discussed in Chap-

ters 20 and 21. Chapter 20 covers the 40-column outputs and Chapter 21 covers the 80-column outputs.

No Sound

There are separate sound circuits in the C128. They are covered in Chapter 22. The sound circuits are mostly contained in the 6581 *Sound Interface*

Device, called SID. When sound troubles happen, they are usually related to that chip and its adjoining circuits.

SID, and the microprocessor it is operating with, form a bond between them. It's as if they are a small computer system of their own. This makes sound troubles relatively easy to test and handle. Sound testing is discussed in greater detail later in this chapter. For all the details though, turn to Chapter 22.

Peripheral Device Problems

As you know the C128 mode is the centerpiece of the computer system. Feeding into the C128 are the keyboard, disk drives, cassette, joysticks, paddles, light pens, inscription pads and so on. The computer processes these inputs and then outputs to the printers, disks, cassette, modem, TV display, etc.

As mentioned earlier, when a peripheral device stops operating normally and causes trouble, the first step is to substitute the troubled peripheral with one that is known to be good. If the trouble ceases and normal computing continues, then the trouble is in the peripheral. Repairing peripherals is a separate job and is covered in other books. However, when the new peripheral also won't work, then the trouble is in the C128.

When you decide that the C128 is at fault, turn to Chapter 23, which discusses the inputs and outputs of the C128. There you will find the circuits and plugs that receive the inputs and send out the outputs. The chapter guides you to specific circuits, as found in Chapter 19. In Chapter 19 are discussions on the 6526 *Complex Interface Adapter* (CIA) chips. These two chips handle most of the I/O (input/output) work that is required for all the peripherals except the ones that produce the video displays and the audio output.

DIAGNOSTIC PROGRAMMING

Once you get into this book, you'll progress through the following steps.

- (1) You'll get an idea of what the printboard looks like as you take it apart with directions in Chapter 2.

- (2) You will become familiar with the location of the chips as described in Chapter 3.
- (3) Chapters 5 through 9 will give you a good idea of what the C128 is doing.
- (4) You will be briefed enough to intelligently perform a lot of your own tests using PEEK and POKE in BASIC, or the machine language monitor commands: MEMORY and FILL (M and F), which perform the same jobs.

PEEK and POKE

Using the function PEEK and the statement POKE gives the C128 the opportunity to test itself and then tell you about it. It is an excellent self-test and works well as long as the C128 is not down completely. Should there be a power supply problem, or other major problem, then the computer obviously can't test itself. However, if the machine is operating somewhat, but is exhibiting glitches or other erratic behavior, then you can often pinpoint the circuit and chip that is defective with these test techniques. If you have been doing any programming at all in BASIC, then you are well prepared to write a tiny test program, run it, and come to diagnostic conclusions.

PEEK allows you to read the contents of any of the locations in the C128 or C64 memory map. This includes all the ROMs as well as the RAM (Random Access Memory) chips, and includes addresses on the map that are in registers on large chips like VIC, SID, and the CIAs. As long as the location is on the map, you can read its decimal contents which in turn can be converted to hexadecimal and binary.

POKE permits you to load a byte or bits into all of the memory addresses, with the exception of the read-only locations (ROM). POKE gives you the ability to run test data back and forth between the processor and the residents of the memory map.

Two ways might be used to get the PEEK and POKE tests to operate. First, you can use them directly without program numbers. Simply type in the desired program line, then press RETURN. If the line orders a PEEK, then the addressed location will yield its contents and PRINT on the display the decimal number equal to the binary contents of the location. Should the line command a POKE, then the binary equivalent of the decimal number you put

in the line will be put in the location addressed by the line. It's really easy because PEEK is, in reality, a "read" operation and POKE is actually a "write." In the direct mode (e.g., LIST or RUN) these operations are performed without further ado.

Otherwise, you could write a tiny test program with numbered lines. You could, for example, write to some locations, install numbers in the locations with POKEs, and then read those locations with PEEKs. Another common test program could POKE numbers into registers to see if the registers are performing. When you POKE registers on a chip, it is exactly like working the switches of a control board of the chip.

When you use POKE and PEEK as a signal injector and location tester in the C128, the only tricky

part is having a clear idea of the relationship between decimal numbers and the set of bits in a byte that the decimal numbers represent. When you POKE a number into memory, you type a number that is code for a particular set of digital bits. As you PEEK a memory location, a decimal number is returned to you and is PRINTed on the screen. The decimal number is the code for the set of eight bits that are contained in the register you have just read.

For example, if you are in the C64 mode, then you could desire, for troubleshooting reasons, to learn what the contents of address number 209 are. It's easy. Simply type in: PRINT PEEK (209). As seen in Fig. 1-10, the answer 36 is PRINTed on the display below your command line. This 36 is decimal for the binary bits 00100100. The bits could also

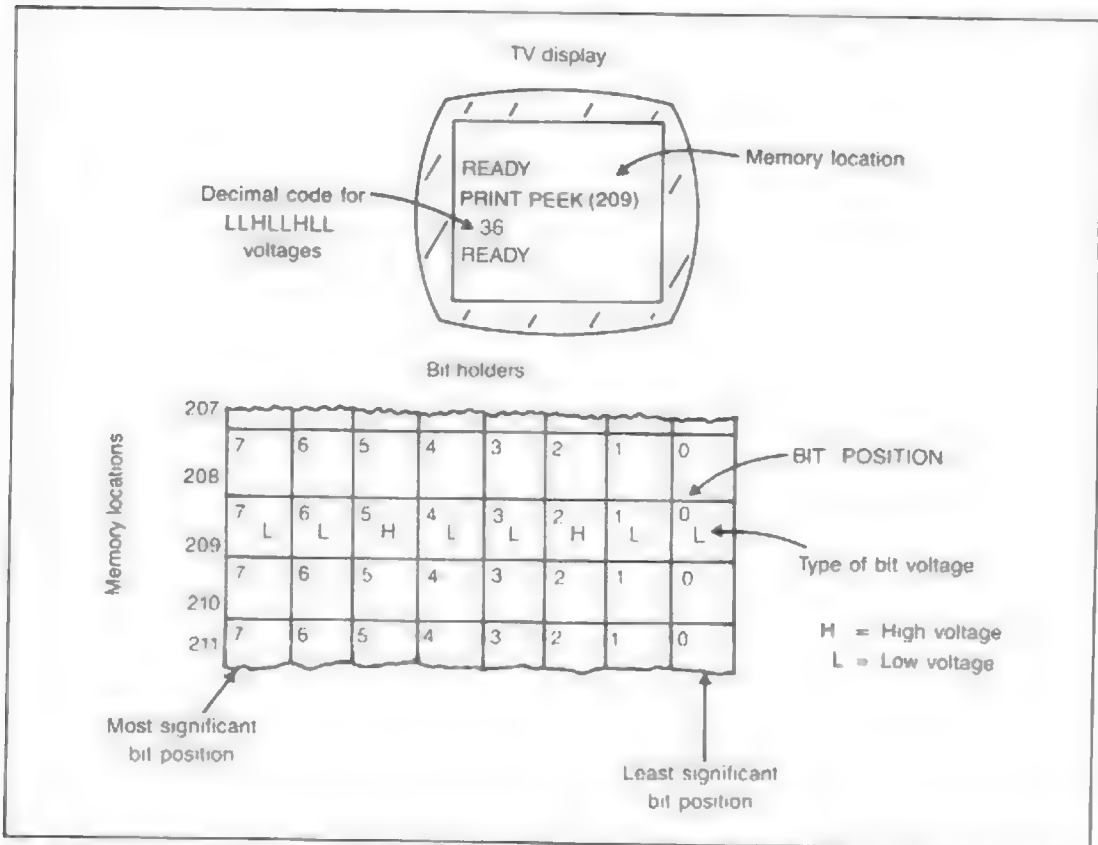


Fig 1-10 A quick-check of a chip is to read the contents of one of its registers. In this test, RAM register 209 in C64 mode is read with a PEEK. When decimal number 36 is returned, the register is considered okay.

be called LLHLLHLL. The programmer usually thinks of the bits as 1's and 0's, while the electronic troubleshooter sees the bits as H's and L's (Highs and Lows). Anyway, the PEEK function can easily read the contents of location 209 while the C128 is used as a C64. A programmer might need the binary arrangement to switch data around in the program. A troubleshooter could use the information to compare the actual contents of 209 to what is supposed to be in 209 at that time. If the correct bits are present, then the location is considered okay. Should the bits be missing or incorrect, then that is a clue that could lead to the pinpointing of a fault.

Another example of using these techniques also refers to the C128 acting as a C64. Suppose that you want to quick-check the operation of the color RAM chip. A fast test could be a change of border color on the display. Location 53280 controls the color of the border. It so happens that if you write the decimal number 8 to that location, the border will change to orange—if the chip is okay. You can enter `POKE 53280,8` onto the keyboard. As you hit RETURN, the TV picture border should change from light blue to orange. If it does, then the quick-check infers that the color RAM chip is okay. Should the POKE produce no effect, or the wrong effect, then the chip or its associated circuits are indicated to be in trouble. This is only a quick-check. The inference is that if the chip responds well to one sure test then odds are good that the chip is okay. There is always the possibility that there could be some other subtle form of trouble that the test is not revealing. However, when a test like this is made and works out okay, chances are very good that the entire chip is okay.

The 53280 is the decimal address of the color RAM register. The decimal eight is a set of bits, called 00001000 or LLLHLLL. The color RAM uses the four lowest bits, HLLL, to switch the border color to orange.

PEEK and POKE can be used as a location tester or signal injector. BASIC is used in both the C128 and C64 modes. It is not used in the CP/M mode. These easy tests are not readily available in CP/M as they are in BASIC. CP/M and BASIC are not permitted to run at the same time.

When in C128 or C64 mode you'll find that the PEEK function and the POKE statements can be used over the entire 128K RAM that the C128 uses, and the full 64K RAM that the C64 has privy to. In addition, POKE and PEEK can be applied to the various static RAM chips, the ROMs, the I/O chips as well as the addresses on the video and sound chips. You should consider the locations on the memory map as bit holders and the decimal data you read out of the locations and write to the locations as code for the bits. There will be more on these techniques as you go through the book.

Diagnostic Programs

Besides reading and writing to individual addresses with PEEK and POKE directly, you can, once you acquire a troubleshooting point of view, also write programs that perform batteries of tests and check out large portions of the memory map. One useful diagnostic programming technique is to place PEEK and POKE in loops. A POKE in a loop can make the C128 able to write to a large group of memory locations automatically, one after the other. A PEEK in a loop could have the C128 read many memory locations and print the results in decimal on the screen.

For example, a diagnostic program could be used to test the ability of memory locations to contain data, while at the same time checking out bus lines for continuity. You would first write program lines that POKE data into locations. Then you write PEEKs to check and see whether the POKEs ever arrived and were installed okay. If all the locations you wrote to are holding the data securely, then they and the bus lines connected to them are considered okay. Should some or all of the data not have reached their destinations, you can then trace out the path they were to have taken. Some sort of problem stopped the data flow. You have diagnosed a suspect circuit area with your test program. There will be more on these techniques in Chapters 14, 15, 16 and 18.

In addition to writing your own test programs, there are diagnostic programs available. Check with your local software store for the names of ones that are for sale. For example, there is one diagnostic

that has been around quite awhile for the C64. You can use it for the C128 when it is in the C64 mode. It is called "64 Doctor." It is manufactured by:

Computer Software Associates
50 Teed Dr.
Randolph, MA 02368

Commercial diagnostics are normally fancier than the ones you will produce—they are prettied up with graphics. Once you load the diagnostic into the machine, then a menu pops up. One option checks out the disk system and the C64 mode's internal RAM. It is limited in that it can only test 64K of the 128K in your machine. Next, there are tests for the keyboard, a printer (if it is connected), a cassette (if it is connected) and joysticks. Another test runs patterns to check out your color TV or monitor. Lastly, the program makes the SID chip perform some music.

This diagnostic also works out the graphics on the C64 mode. It produces sprites that look like a TV set, a printer and so on. The sprites march around the display.

For troubleshooting, a program like this has use but is limited. First of all, if the computer is down completely, then it can't run any program, including the diagnostic. However, the program can sometimes decide whether a peripheral problem is located in the peripheral itself, or if the computer circuits are not performing properly. Should you be unable to substitute a peripheral that is known to be good for a suspect peripheral, as a test, this program could help.

While these programs are fun and occasionally useful during troubleshooting, they can be valuable before you begin a programming session. Should you be planning a long program, it is a good idea to exercise the C128 everyday before you put program lines into the machine. If the machine exercises okay, then it is safe to work on. It is very frustrating to spend hours programming, and then discover there is some fault with the computer.

PRINTBOARD LANDMARKS

A C128 user is mostly concerned with software and applications. However, when trouble strikes,

that focus must be changed. You must switch your view from the software to the hardware that is running the software. The next chapter describes how to open up the C128 and get the printboard out in the open. Once the printboard is exposed, you will see a complex maze of chips, capacitors, resistors, foil connecting lines and many other items. In order to make any sense out of the layout, you have to find out what all those components and connections are. The place to start is with the landmarks. Once you recognize them, you'll start to get to know your way around.

The main landmarks are the large chips, the group of ROMs, and the collection of 16 RAMs. A close look reveals the data and address buses coursing over the board. Figure 1-11 depicts the nine large chips, the locations of the ROMs, and the RAMs.

Figure 9-1 is a block diagram of how these chips are connected electronically, and the directions that the data takes. The data flows over the system data bus. The data enters the circuits via the keyboard into a CIA, and passes onto one of the MPUs. The MPU then reads the ROMs, reads and writes to the RAMs, and then outputs to one of the video chips, a second CIA and to SID. The video chips in turn output to the display and SID outputs to a sound system.

Figure 1-12 shows how the address bus performs, and how the chips are involved. The addressing emanates from the MPUs. They send the address signals to the PLA (Programmable Logic Array) and the MMU (Memory Management Unit). These two chips then form the various addresses and they are able to contact all the residents of the memory map. When a memory location is addressed by this system, then data can be sent to its bit holders.

Complex Interface Adapters

The two CIAs, U1 and U4, are physically located on the two sides of the printboard. U1 is found in the lower right-hand corner and U4 is on the edge of the left side just above the center of the board. Both chips are plugged into 40-pin sockets. They are both numbered 6526 and are called CIAs for Complex Interface Adapters. The right side one is

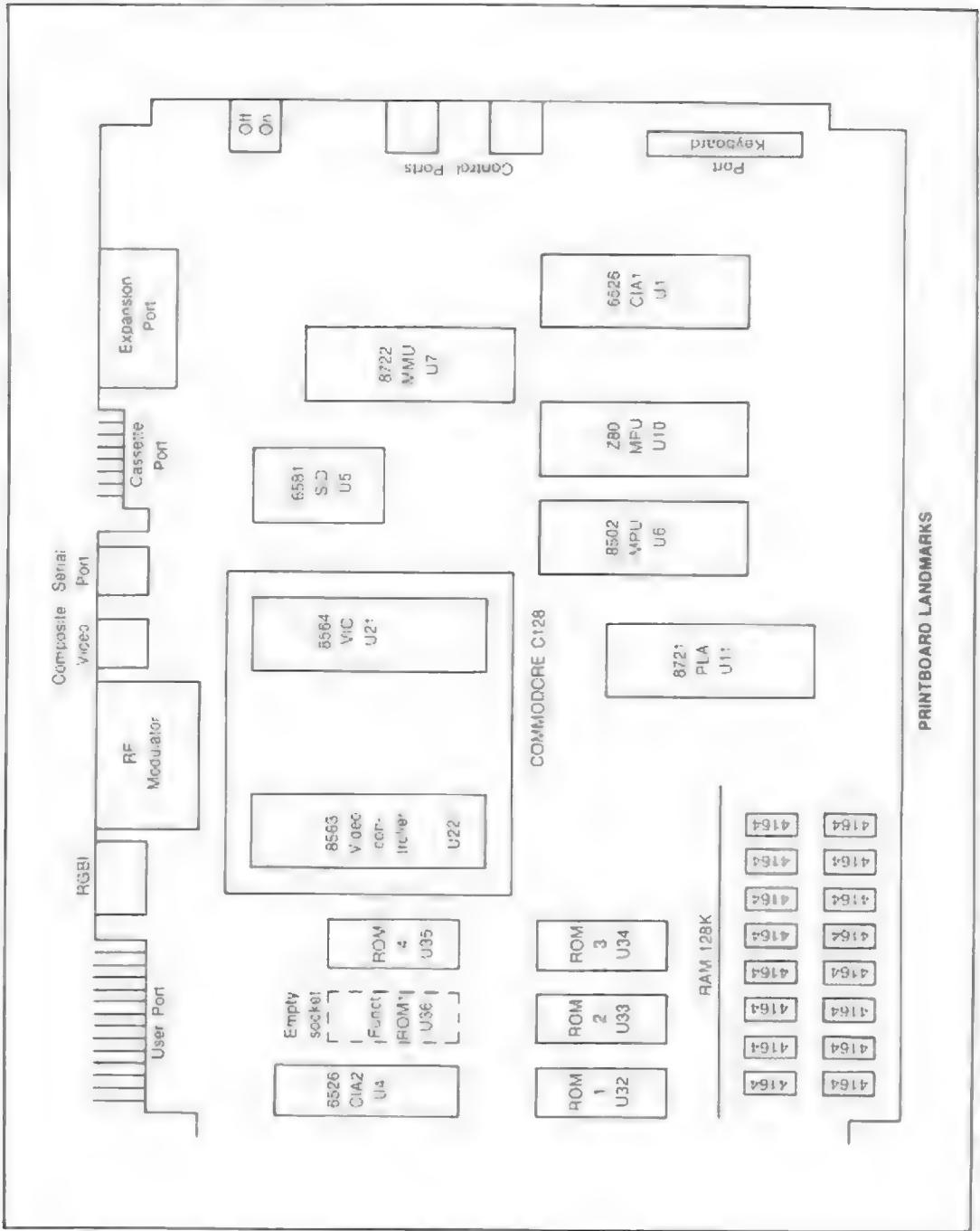


Fig. 1-11 When the C128 is opened the landmarks of the printboard can be seen. The landmarks are the large chips, the RAM and ROM sets, the metal boxes and the various parts.

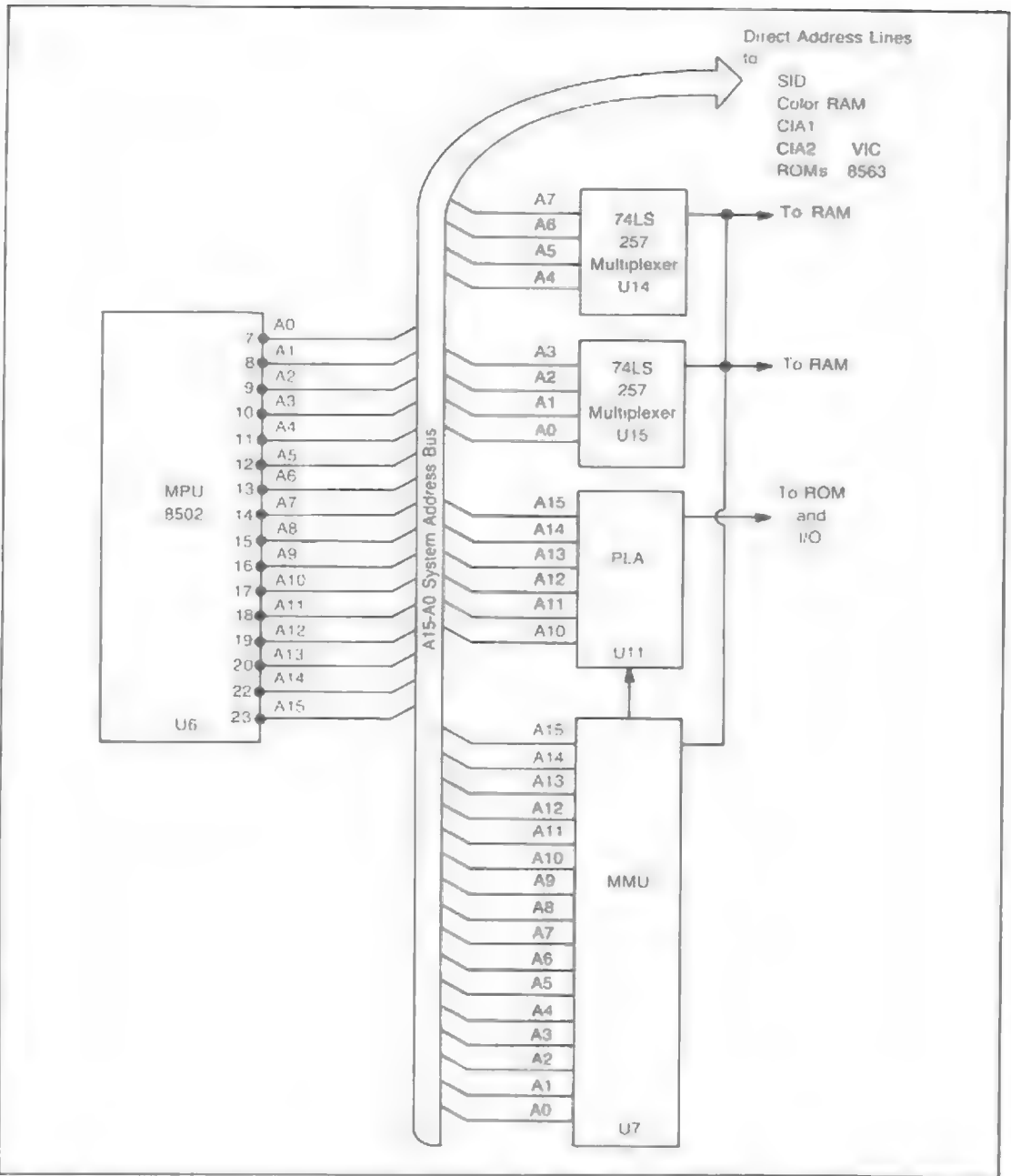


Fig 1-12 The 16 lines of the addressing system originate in the MPU. They are named A0 through A15. The lines go to all the residents of the memory map. The 16 lines are able to address 65,536 individual register locations.

wired to a plug that the keyboard is connected to. This is the keyboard's port of entry.

The right side CIA, called "CIA1" on the schematic, besides letting the keyboard pulses in, also is the port of entry for the two input plugs the joysticks and other peripherals use. When a key is struck or a joystick is moved, the electronic signal generated enters U1, the CIA1. The signal is processed by CIA1 and is then output to the eight parallel copper traces called the "System Data Bus." The data bus then acts as the pathway for the signals to connect up to the microprocessor further down the board.

U4, called "CIA2," on the left side of the board, is the port of entry for signals entering or leaving the User Port and the Serial Socket. CIA2 performs in the same manner as CIA1.

When I/O troubles crop up, an easy analysis can clue you into the circuit area that could possibly be containing the fault. This is shown in Table 1-1. For instance, troubles with keyboard or control port inputs could be originating in U1, the CIA1. Problems with the input or output of the User Port or the Serial Port could be a CIA2 circuit flaw. Chapter 23 goes into details on this I/O subject.

The Microprocessors

There are two microprocessors in the C128. The main processor is U6, an 8502. It is an upgrade of the 6510 processor found in the C64, which in turn is an upgrade of the 6502 that resided in the VIC 20 computer. All three processors use the same Instruction Set.

The 8502 sits in the lower right quadrant of the printboard. It is in the center of the circuit. It is the originator of the data bus and the address bus. It does not have addresses on the memory map. It can be likened to a central telephone exchange—the locations on the memory map are the telephone numbers that the exchange services. The CIAs have addresses, and when the 8502 dials them up they respond over the data bus. While the address bus is only one-way from the 8502 to the addressed location, the data bus is two-way and can be read from, or written to, by the 8502.

The 8502 is used when you put the C128 into the C128 or C64 modes. Besides connecting to the CIAs, the data bus from the 8502 hooks up to all the rest of the memory map locations. This is discussed in detail in Chapter 18.

U10, the Z80 microprocessor, is called the "Coprocessor." It is a full-fledged processor on its own. But since the 8502 handles the C128 and C64 chores, the Z80 is given a subservient role. The Z80 conducts the CP/M activities of the machine. It is located on the board, next to the 8502, on the right. Both processors have 40 pins. However, the 8502 is plugged into chip sockets but the Z80 is not. The Z80 is soldered directly to the printboard.

The two processors work independently of each other. When the 8502 is operating, the Z80 is disabled and sits quietly. As the Z80 takes over for the CP/M operations, the 8502 is turned off and just waits idly by. With the two separate processors, the C128 total machine thus becomes a form of Siamese twin. The twins are joined at the I/Os and memory. Furthermore, since the 8502 is conducting both C128 and C64 operations separately from each other, a third machine is present making the C128 package Siamese triplets.

When trouble strikes a processor, the function it is executing will fail. If the 8502 gets sick or dies, the C128 and C64 operations will be the ones to suffer. When the Z80 passes away, the CP/M operations won't work. The 8502 and its problems are covered in Chapter 12. The Z80 processor is discussed in Chapter 13. There is more about both of them in Chapter 5.

The Addressing Management Chips

When one of the processors dials up an address, all of the bits do not travel directly to the desired location. Some of them are routed to two substitution chips. They are U11, the 8721 PLA and U7, the 8722 MMU. The PLA, U11, is a 48-pin chip and is soldered at center bottom of the printboard. The MMU, U7, is also a 48-pin chip and is located in a socket on the right-hand side of the board about half-way down. The PLA is covered in detail in Chapter 14. PLA stands for Programmed Logic Array. The

MMU is discussed in Chapter 15. MMU is short for Memory Management Unit.

The two large chips work together to keep the addressing straight. With two processors, five various operating modes, two separate type video output chips, 128K of memory in two 64K banks and many other complications, these chips have their registers full.

When trouble strikes in these chips or their circuits, the main symptom is bad addressing, which can result in garbage or other related symptoms.

The Video Output Chips

In the upper left quadrant, near the center of the printboard is a metal shielded enclosure. Inside the enclosure, on the right, is a 48-pin VIC chip soldered to the board. It is U21 and named the 8564. It is an upgrade of the VIC chip found in the C64. VIC stands for Video Interface Chip.

On the left-hand side of the enclosure is another 48-pin chip plugged into a socket on the board. It is U22, the 8563. It is the Video Controller.

These two chips are quite complex and are sometimes referred to as microprocessors. They can do almost the same job as a processor. VIC chips can be used in video game machines as a processor and also as an I/O chip.

The VIC 8564 is the subject of Chapter 20 while the 8563 video chip details are in Chapter 21. These two chips do not work together. Like the 8502 and the Z80, when one is on the other one is off. The VIC chip is used exclusively for all 40-column operations. This includes the C128 40-column mode, the C64 40-column mode and the CP/M 40-column mode. The 8563 Video Controller is used exclusively for all 80-column operations. This includes the C128 80-column mode and the CP/M 80-column mode. The C64 does not have an 80-column mode.

The VIC 8564 outputs directly to both the RF Modulator box at the top of the printboard and to the Composite TV Video plug to the right of the RF Modulator box. These are the exclusive 40-column outputs. The 8563 video controller chip outputs to the RGBI output plug only. This plug puts out only 80-column video signals.

The Sound Interface Device

Sitting in a socket to the right of the board's center, in the top half of the board, is SID, the 6581 audio producer. SID is only a 28-pin chip and has little to do with any of the circuits except for the microprocessors. SID takes care of all the sound requirements of the C128. It is connected to address and data bus lines. SID is covered in detail in Chapter 22. When sound troubles occur, SID is the prime suspect.

SID outputs its audio to both the RF Modulator and to a couple of pins on the Composite Video plug. The RF Modulator has audio from SID sent into its audio input plug. The Composite Video connector devotes two pins to audio. Pin 3 handles the audio output from SID. However, pin 5 takes care of any audio input that comes from an external device. The outside audio is then sent to SID where it can enter the chip and be processed according to SID's dictates.

AN OVERVIEW OF TROUBLE ANALYSIS

When trouble occurs with your C128 system, the first step, as discussed earlier in this chapter, is to determine whether the trouble is in the computer or in a peripheral. When the trouble is in a peripheral, you have the peripheral fixed. Should the trouble be in the C128 itself, then this book applies.

Figure 1-13 is a flow chart that could help you decide the general circuit area on which you should focus. Before you start on the flow chart, if the computer appears dead and the indicator light is out, go no further—the trouble is in the power supply. Chapter 24 contains the service information for that condition.

Should the light be on, then the flow chart can begin. Start by analyzing the symptom. The purpose of the analysis is to decide on what service approach to take. For example, if the usual sign on display is completely gone, and you know the TV is okay, then the computer is not putting out any video signal in that mode. What you must do is figure out if all of the computer signal is gone, or just the video.

A quick test could be to write a test program to SID and determine if SID is outputting. There is

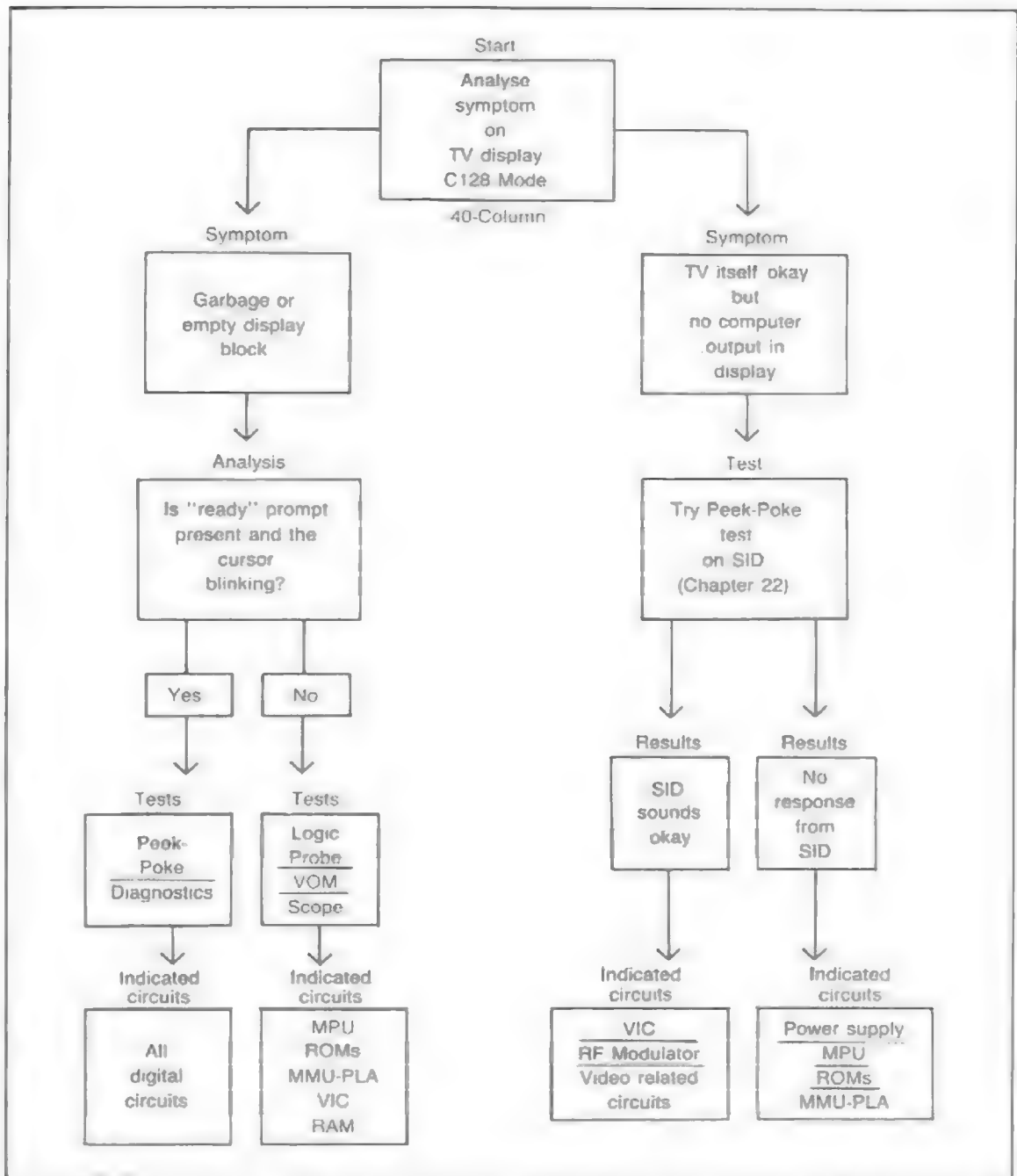


Fig 1-13 When trouble strikes, a good starting procedure will save you from wasting time. This is an example of the way to go from symptom to pinpointing the defect.

a small PEEK and POKE program in Chapter 22 that will do this. If SID remains quiet, then there isn't any computer output, sound or video. This indicates power supply trouble even though the indicator light is on. Chapter 24 has the test procedures.

On the other hand, if SID does start making test tones then the computer is putting out some signals. By the process of elimination, the video circuits are the prime suspects. Look over the video chapters, 20 and 21.

The other general symptom is: some sort of display, but not a useful one. There could be garbage on the screen, an empty display block, a display that has locked up and won't respond to keyboard

strikes, or just erratic operations. You could, in those cases: try the reset button; turn the computer off and on; depress the RUN/STOP and RESTORE keys at the same time. If these measures do not cure the problem, then examine the READY prompt and the cursor. Are they present and is the cursor flashing? If so, you might be able to use PEEK and POKE tests or a diagnostic program you have.

When the READY prompt and the cursor are disabled, then you probably can't get the computer to cure itself. At that point you must resort to the various logic probe, vom and scope tests as described in the book.

2. Disassembly

When a C128 gives up the ghost and you make a decision to repair it, then most of the time you must take it apart. The first time you start taking out the screws, you are sure to be hesitant. You know from past experience with many other repair jobs that you could just possibly cause some additional troubles by simply taking it apart. You don't want to start a repair job by causing trouble.

Fortunately, the C128 is assembled in a sensible manner, which makes the disassembly relatively easy, although extreme care and slow moves are the order of the day. The first steps are common sense. Arrange a large enough place for your work. Gather your tools together. Be sure to have good lighting and place a rubber mat on the bench. Disconnect all the attachments to the C128. Place it on the soft mat. Figure 2-1 illustrates progress to that point.

It is a good idea to have the bench area as clean as possible. Dirt and filings that accidentally get into the computer could end up as additional troubles. It is not a good idea to work on the C128 in a low humidity environment. Should it be cold outside but

warm and dry inside, with static sparks flying and popping as you walk on the carpet, be extra careful while working on the computer. Static electricity could be death to computer chips. There will be more about the static electricity precautions in Chapter 4.

GETTING THE HOOD UP

The C128 is put together as a sandwich. The top slice holds the keyboard, the bottom slice holds the printboard, and the components are connected to the printboard. The first thing to do is take the C128, turn it upside down and place it on the soft pad, keyboard on the bottom. Then check the screws. There are six of them. They are located as shown on Fig. 2-2.

The six screws on my model are tiny little jeweler's types with a number 10 Torx hole instead of a screwdriver slot. You should have the correct form of driver to remove the screws. I purchased my number 10 Torx screwdriver at Sears for \$3.99 plus tax. With the correct screwdriver, the little ones



Fig. 2-1. To disassemble the C128, you need a number 10 Torx screwdriver, a Phillips head screwdriver, long nose pliers and a low-wattage soldering iron. For the reassembly, a tube of heat-transferring silicone paste should also be used. The logic probe, chip straightener-inserter and small cutters are also useful during troubleshooting.

come right out, as in Fig. 2-3. If you should not have the Torx driver, you'll have a hard time removing the screws.

Once the six screws are out, then place them all together in a safe place so that they will all be available for replacement. Grasp the C128 so that

it doesn't open and turn it back so that the keyboard is on top once again, then try to pry the top away from the bottom.

The case is plastic and the top edge is seated in little holders. Sometimes the edges get wedged in the holders and it appears difficult to separate the

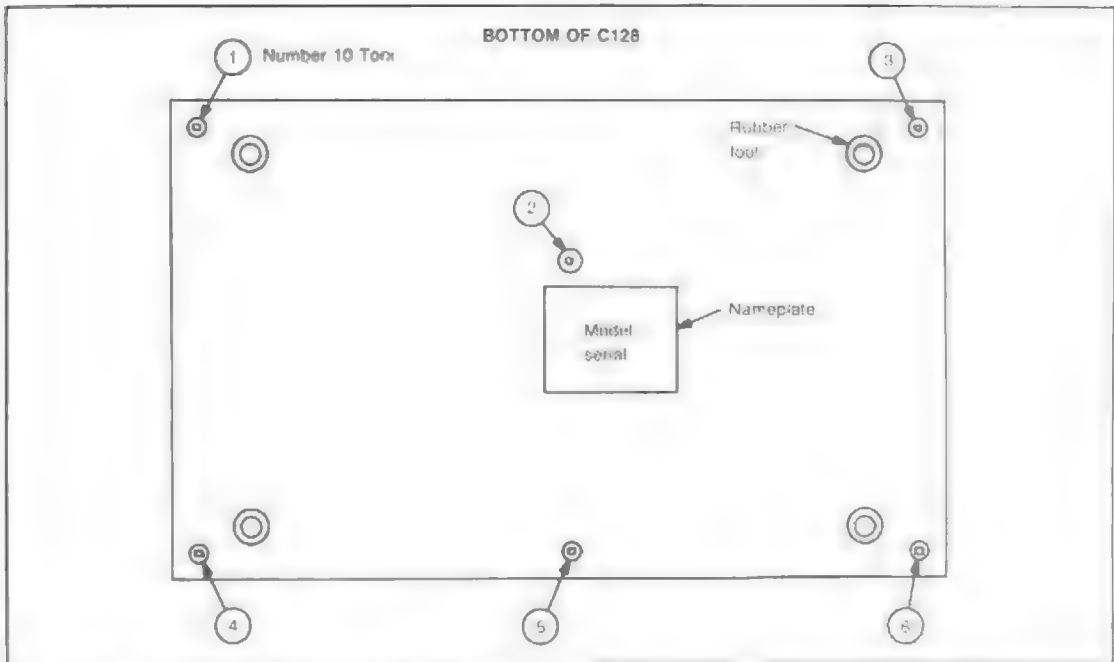


Fig. 2-2. On the bottom of the C128 are six Torx screws located at these spots.

top of the case from the bottom. Be persistent and keep prying, gingerly. The top will dislodge itself and separate after some prying.

Once you separate the top and bottom, then lift the left side up first. The power indicator is on the left side. A three wire socket on the printboard holds the indicator three wire plug. Disconnect the plug, as in Fig. 2-4. Raise the top of the case and the top will swing up from left to right. The reason that the top will not come straight up is another number 10 Torx screw, seen in Fig. 2-5, on the lower right side of the printboard that is holding a ground strap connected to the keyboard. Then you will see that the keyboard is plugged into a 25-pin socket on the main board. Unplug the keyboard, as in Fig. 2-6, and the top of the case is free.

Unless the keyboard itself has troubles, there is no reason to remove it from the top casing. However, should you need to remove the keyboard from the top, it is easily taken off by removing six more number 10 Torx screws. The keyboard will then be free.

FREED THE PRINTBOARD

Once the top is off, you'll be looking down at a metal shield that covers the entire printboard, top and bottom. The shield is ventilated with a lot of holes, but it covers the entire board. In order to do any testing or parts replacing on the main board, this shield must come off.

The first step is to remove the seven number 10 Torx screws, in Fig. 2-7, that are securing the printboard-shield assembly to the bottom of the cabinet. The printboard and shield assembly will then lift right out of the cabinet bottom, as Fig. 2-8 shows. The next step is to separate the printboard from the shield.

A close examination of Fig. 2-9 shows that the shield has eleven twist tabs hooking the top and bottom parts of the shield together. The tabs are located on the front and sides of the shield. These tabs must all be straightened. Once the tabs are no longer holding the top and bottom of the shield together, there is one last connection and one more screw. The connection is a solder spot next to a twist tab

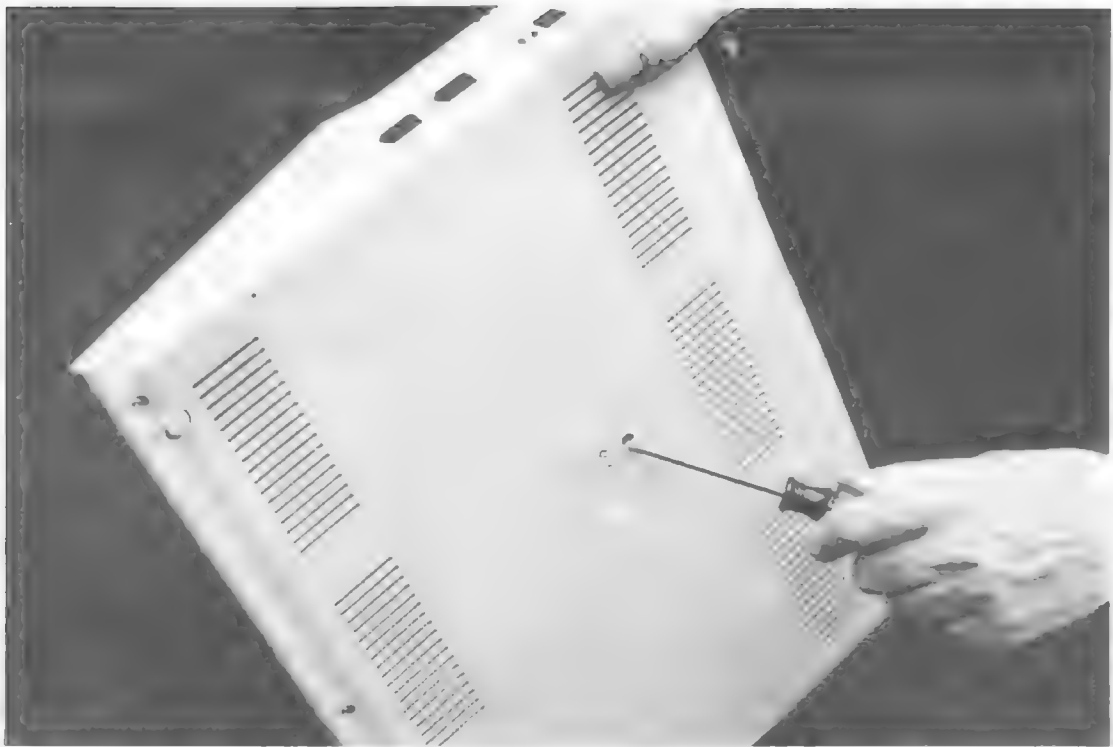


Fig. 2-3. The six Torx screws are removed easily. Don't forget the hidden center screw above the nameplate

holding the shield to the printboard, as seen in Fig. 2-10. It is located on the right side of the board. After you desolder that connection, there is a Phillips head screw, shown in Fig. 2-11, located on the top of the shield right in front of the RF Modulator box. Once that screw is out, then the shield can be removed, top and bottom, as seen in Fig. 2-12.

Once the shielding is removed, then the printboard is revealed. You'll see 55 of the C128's inventory of 63 chips on the board. Some are plugged into sockets while others are soldered directly to the board without the convenience of sockets. If you need to change a chip in a socket, it is a ticklish job to make sure the pins are lined up properly, but the job shouldn't take more than a minute or two. Should you be unfortunate enough to have to replace a chip soldered to the board, you have a tedious soldering job ahead fraught with the danger of "inducing" additional problems. Chapter 4 details the techniques

required for replacing chips in sockets or soldered to the board.

In the top left quadrant is the RF Modulator box and the metal shield box containing video associated chips. The top of the video box comes off, as shown in Fig. 2-13, and you'll see eight more chips. You'll find six small chips, U22 (the video controller) on the left and U21 (the VIC) on the right.

When you remove the shielding, note that there is a heatsink that is designed to touch the shield on the VIC. The sink is covered with a white silicon paste to conduct the heat away from VIC. Should you disturb this heat connection be sure to apply fresh silicon paste. You can obtain it in any electronic store, such as Radio Shack.

It is vital that, as you take apart the C128, you perform the disassembly in a slow and careful manner. Note the way it comes apart so you will be able to get it back together again without undue difficulty.

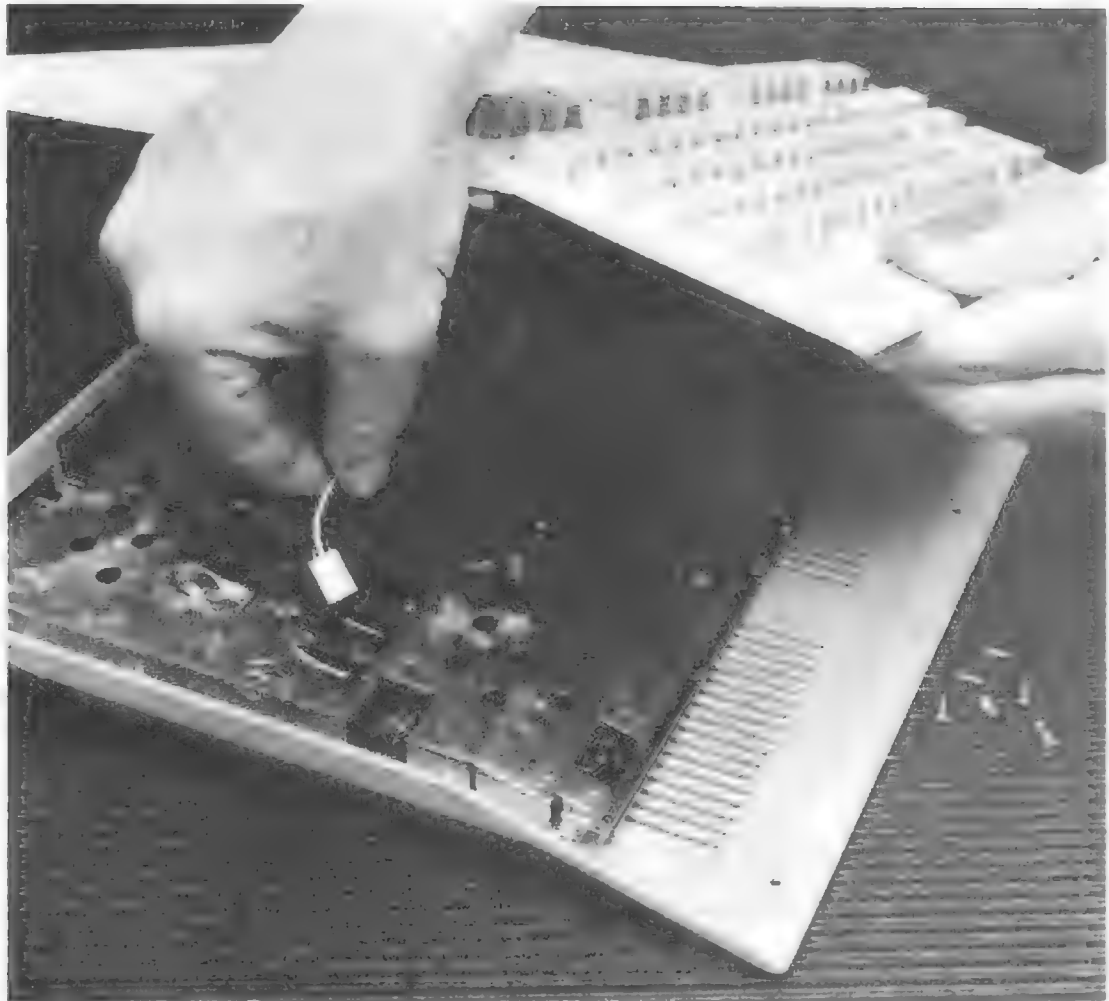


Fig. 2-4. Once the screws are out the top will dislodge itself. Lift the left side and disconnect the LED pilot light plug.

THE POWER SUPPLY BOX

Power supply troubles are among the most common in computers. If the computer is dead, then the first step is to check out the supply box. Chapter 24 goes into detail on the techniques required. If you find that the box is, indeed, the source of the trouble, then the next step is to try and disassemble the box. Sometimes this is an easy task and at other times is almost impossible. This is because there is

no one box that is found with all C128's. They all produce the same supply voltages for the C128, but with slightly different circuits and very different casings.

The last two boxes I encountered had two different casings. One was very heavy, which meant that it was probably potted (a heavy waxlike plastic was poured into the box to seal the circuits in place).

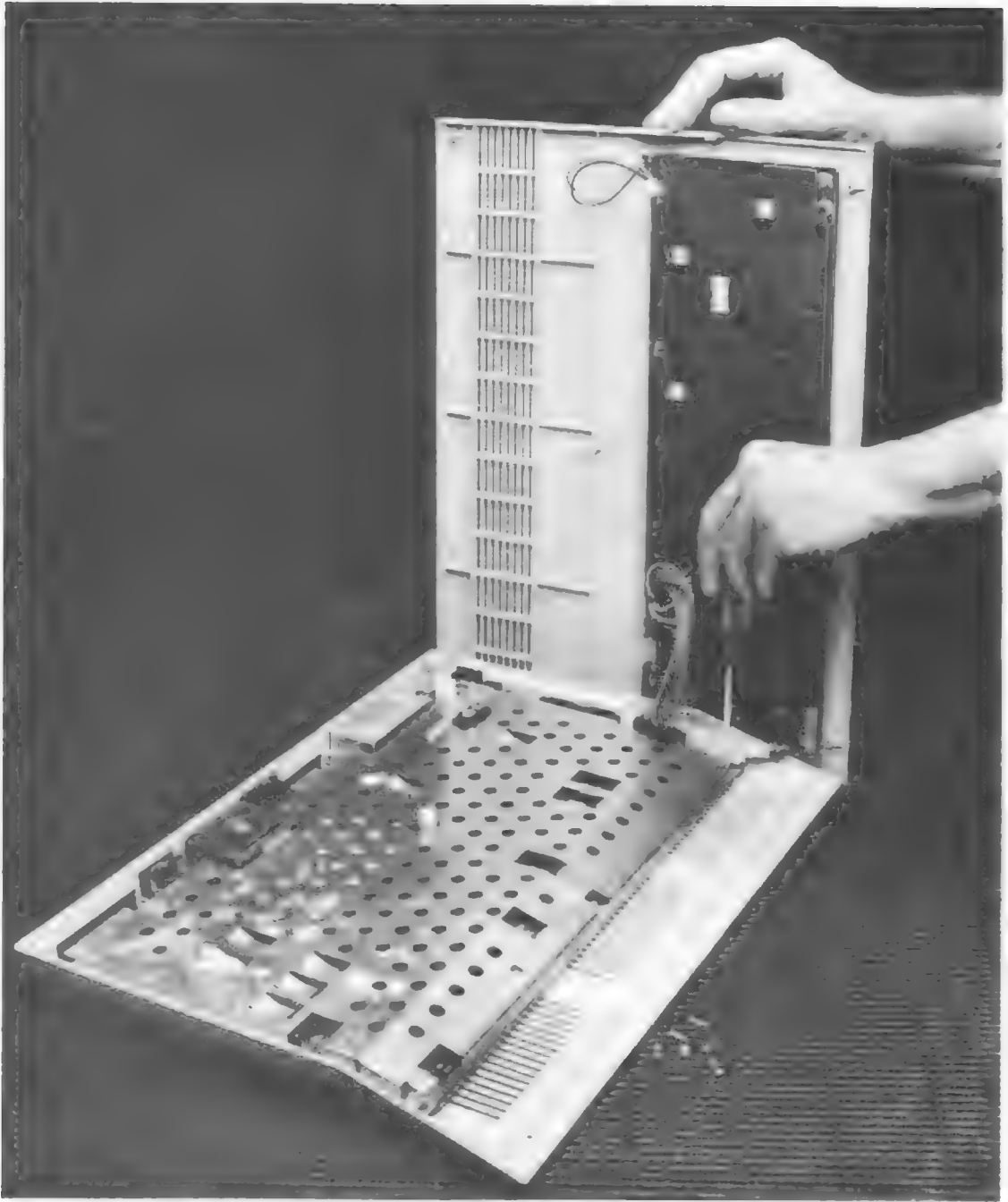


Fig 2-5 Another Torx screw holds the keyboard ground strap to the main chassis.

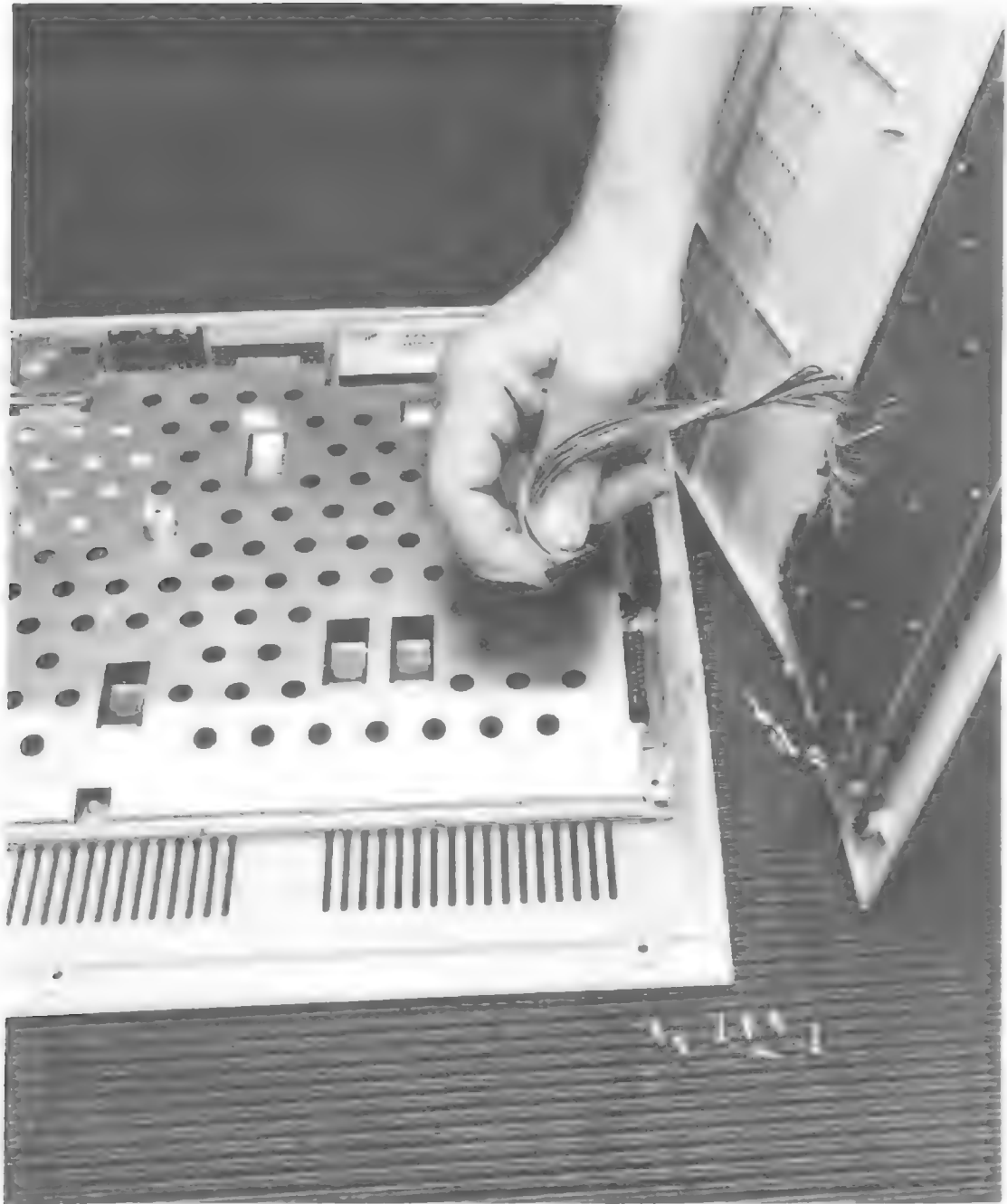


Fig 2-6 The keyboard plugs into the main chassis with this 25-pin plug. It pulls off easily. When reinserting make sure the plug is seated snugly.

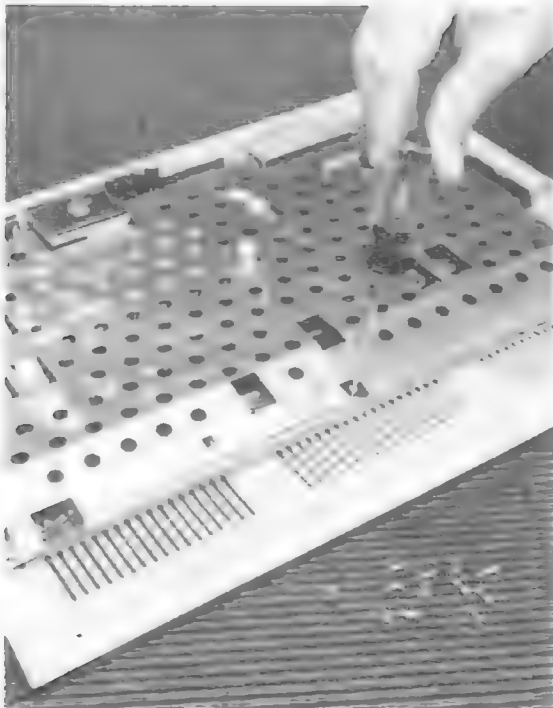


Fig. 2-7. Seven more Torx screws secure the printboard-shield assembly to the bottom of the case.

In spite of the fact that it was potted, I attempted to pull it apart because the trouble was definitely in the box. There was one long bolt. I removed it and tried to pry the top from the bottom of the case. They were so stuck together that the only way they would come apart was if I broke them apart. I opted to simply replace the power supply with a new one. It was the economical thing to do.

On the other hand, another box I worked on was easy. It was much lighter than the first one. The box was not potted. It had four long bolts as shown in Fig. 2-14. Once they were removed with a thin Phillips head screwdriver, then the top lifted right off.

Inside the box is a small printboard and some other circuits, as shown in Fig. 2-15. There are also two small fuses in fuse clips. Replacing a bad fuse might be a quick repair. In addition a power transformer, some diodes, filter capacitors, transistors and one integrated chip are in the box. Chapter 24

details the circuit and the troubleshooting methods needed to fix troubles.

VISUAL REPAIRS

After you disassemble the C128, then you are afforded an excellent easy kind of repair that can produce a repair in a small percentage of cases: some troubles can be seen and promptly remedied. One such kind of trouble is accidentally installed in the computer during manufacturing. After the printboard is assembled, then it is soldered with automatic machinery. During the soldering process, the machinery generates hot gases that can expand rapidly and shoot hot liquid solder into the air. As the solder cools and falls, it hardens. The solder then drizzles down and some of it could find its way onto a newly assembled printboard. The drizzle contains solder flux which can act as glue. Where the solder falls it sticks.

Of course, the factory knows all about this, and the board goes through extensive cleaning in the final stages. However, as hard as they try, an occasional sliver of solder will stick in a place where it frustrates removal, as shown in Fig. 2-16. Also, it does not cause any problems at that time, so it sneaks through quality control, and is shipped out with the finished computer.

The computer and solder sliver travels many miles and is handled by a lot of people and machinery. It finally ends up on a users desk. During the next few months as the computer is used the sliver jiggles around and then manages to lodge between some copper etch lines of the address bus. When the computer is then "fired up," then instead of a normal sign-on picture, a screenful of garbage results. The computer needs service.

If you decide to check it out yourself, then the first thing you do is take it apart. Once the printboard is freed from its shielding, then the next step is a close visual inspection under a good light. If you are alert, then you would sight the sliver of solder and remove it. It could be that the computer is fixed. As you turned it on, the normal sign-on message will appear. All you have to do then is put it back together.

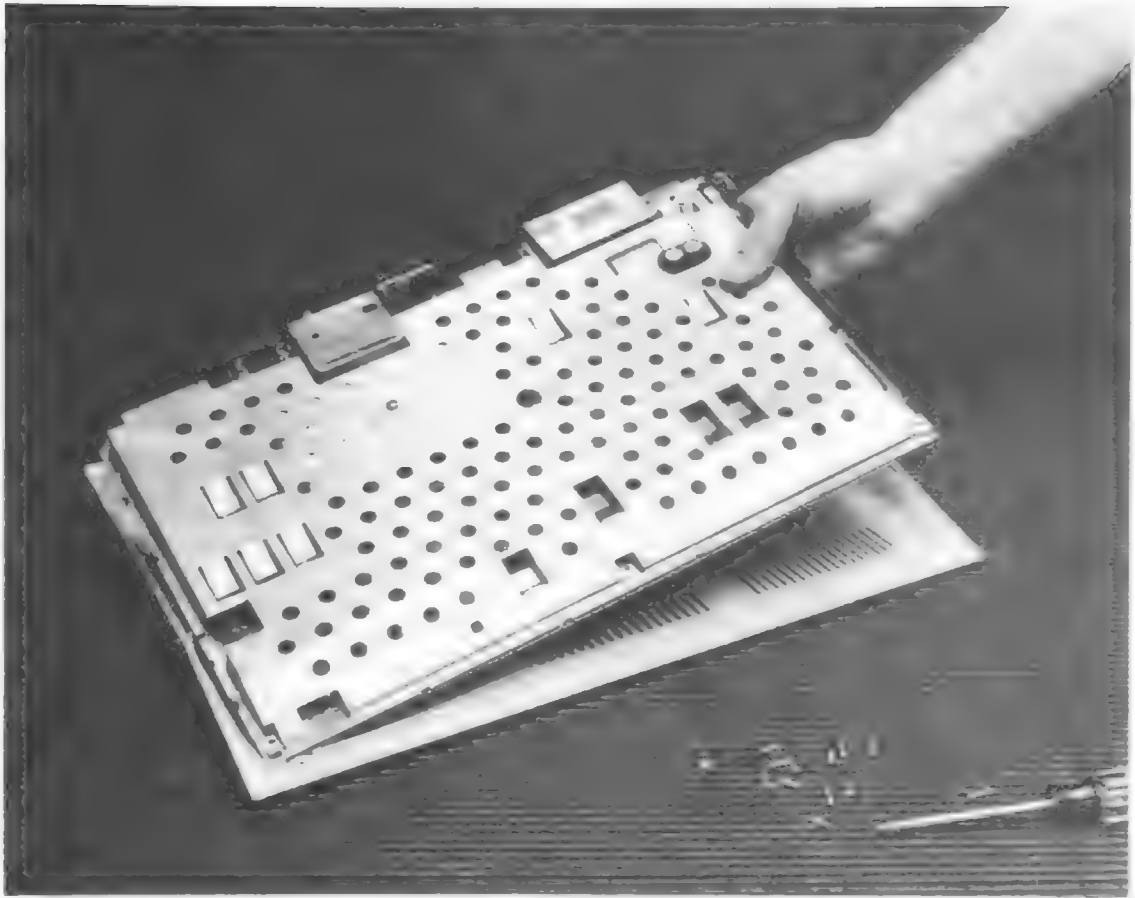


Fig 2-8 The printboard and shield assembly will lift away from the bottom of the case after removing its restraints. Sometimes it is a very snug fit and will require some careful prying to free it

While solder sliver troubles do happen on occasion, most of the time you are not so lucky. There are, however, a few other visual service moves that are easy to make and that often do produce good results. The only pieces of test equipment you need are a bright light and a good magnifying glass. You will be looking for short circuits, open circuits, and burnt or mangled components.

Long Lead Shorts

One common type of short circuit happens when a component's lead, sticking out of the bottom of the board, is left too long and manages to touch an

other part of the board or the shield as shown in Fig. 2-17. The computer is put together in layers. The cabinet is at the very bottom with the shielding sitting on the bottom. The shield acts as a common ground for the entire printboard as well as a shield to ward off and absorb electrical interference that might come through to the C128.

A short of the long lead occurs when one of the printboard's active connections manages to contact other active connections or the ground. You can often spot long leads that are producing a short circuit and snip off the excess or bend them clear of the shorting spot.

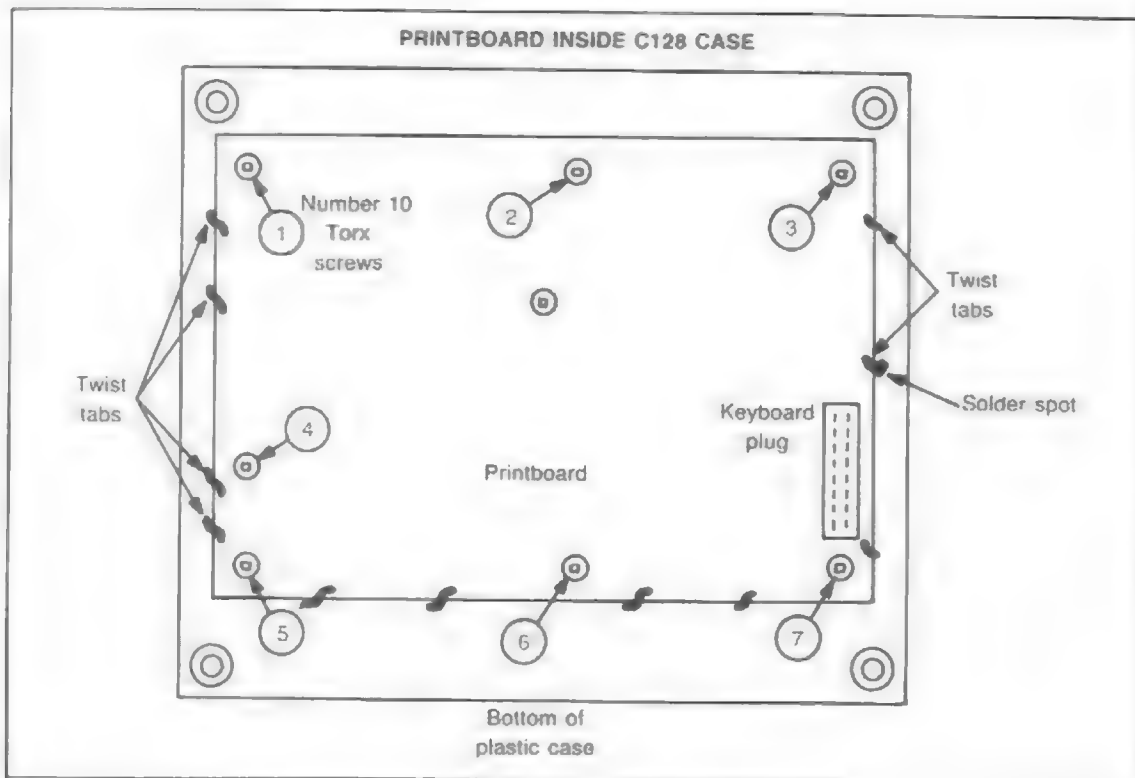


Fig. 2-9 Some 11 twist tabs must be straightened to loosen the top shield from the bottom.

Board Defects

There are thousands of connections, long lengths of copper etch lines, etc., on the printboard. Sometimes a socket or chip pin can become bent under, instead of properly soldered into its board hole. It could also sneak by inspection because it works okay—the socket is otherwise firmly attached and the bent-over pin makes a pressure contact. However, in about a year, some corrosion builds up on the pin and the pressure contact no longer holds. Erratic performance develops leading to a permanent disability.

This type of trouble can be seen with the bright light and magnifying glass. Once you locate it, you can gingerly move the pin into its appointed hole and apply a tiny drop of solder with a small soldering iron.

Other types of board defects can also be found visually. The address bus, the data bus, and the

other bus lines are copper etch lines on the printboard that travel over much of the board. These bus lines must be continuous. There cannot be any breaks or touching between lines. If you find a break or a place where they are touching, then you have located an open or a short. Figure 2-16 shows both problems. Either way, you have trouble. Chapter 18 covers all these bus lines in detail. It is a good practice to examine all the bus lines carefully before each repair. You could just be lucky enough to find a solder sliver or a break and have a quick fix by removing the short, or resoldering the break.

Blackening

Another item of repair importance is blackening. Look carefully at the board for blackened components. If a resistor or capacitor starts smoldering, the chances are good that it will become blackened.

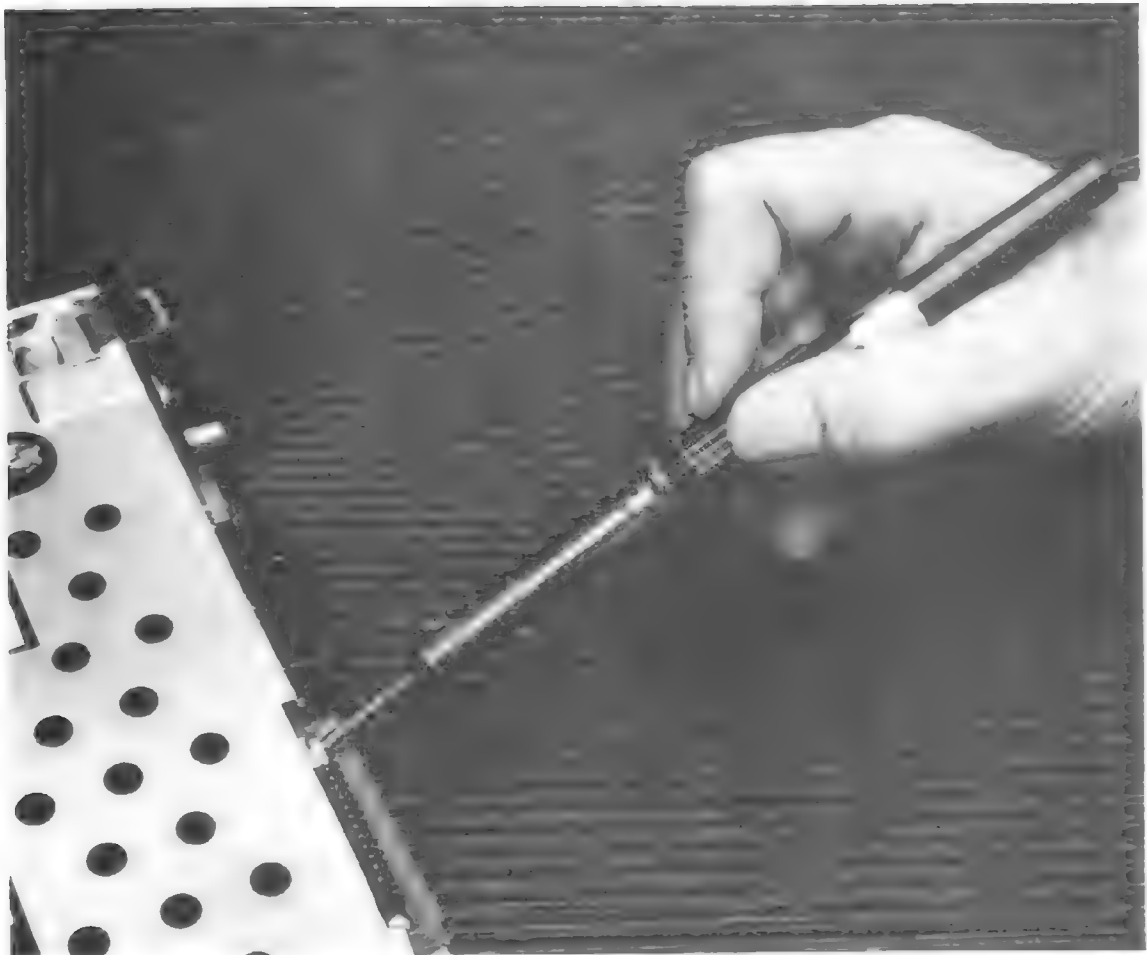


Fig. 2-10. One solder spot holds the top shield to the bottom shield. It must be disconnected

A burnt cover on a component is a sure sign of a bad component. In the same vein, you can often find trouble by feeling the temperature of a component. For example: the power supply box, when operating, becomes a bit warmish to the touch. If it is cold it is not operating. Other examples are the chips. Some of them are designed to operate warm and other ones quite hot.

The Feel Test

Table 2-1 is a chart for the usual feel of the large chips in the C128 with the ranges of "cool,"

"warm," and "hot." If the chip does not feel cool, warm or hot chances are it is defective, according to the chart. The way to perform the feel test is with the printboard freed and running for a few minutes. Turn the computer off and pull the plug before touching the chips.

Besides being sure you do not contact electricity by pulling the plug out before you stick your finger on a chip, you must also be very careful as you touch each chip. Some chips get very hot! They could burn your fingertips. Therefore, touch the chips with great care.

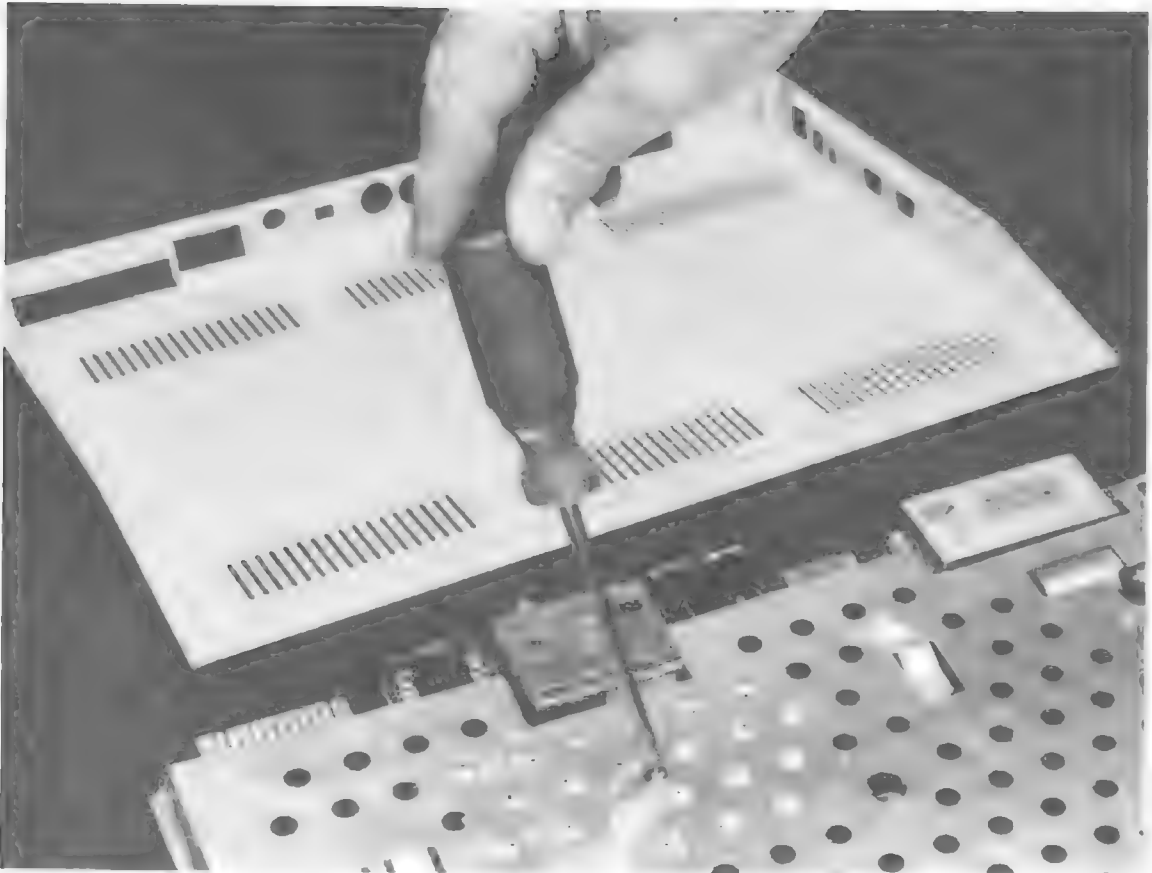


Fig. 2-11. A Phillips head screw holds the top shield to the video case top in front of the RF Modulator box.

CLEANING

Another service move that might complete a repair, but is useful in any case, is cleaning. While the computer is apart, you will be able to see how much dust has been collected. When a computer is used, and time goes by, the inside will collect dust, unless you take extraordinary dust cover precautions. Actually, dust itself is not really an electrical problem. Ordinary dust is an insulator and, as such, won't short out the printboard or its components. The problem with dust is that it blocks off adequate ventilation. Dust enters through the ventilation slots. If enough of it collects, then air circulation is restricted which, in turn, can cause overheating. Should dust get into moving parts, as in the keyboard or

into the interface ports, it could cause clogged plugs, sticky keys and other types of erratic operation.

There are many ways to remove dust. I find the best way is to brush the dust out carefully with a thin, clean, dry paint brush. The dust removal should be done slowly and carefully. Be especially careful around the RAM section and the large chips since they are the most vulnerable to static electricity. Try not to dust on a day that is especially cold and dry where static electricity is jumping off of you. It is a good idea to ground the brush. There is more static electricity grounding information later on in this chapter and in Chapter 4.

Never, ever use any water or other household cleaning solutions on the board. The board must al-

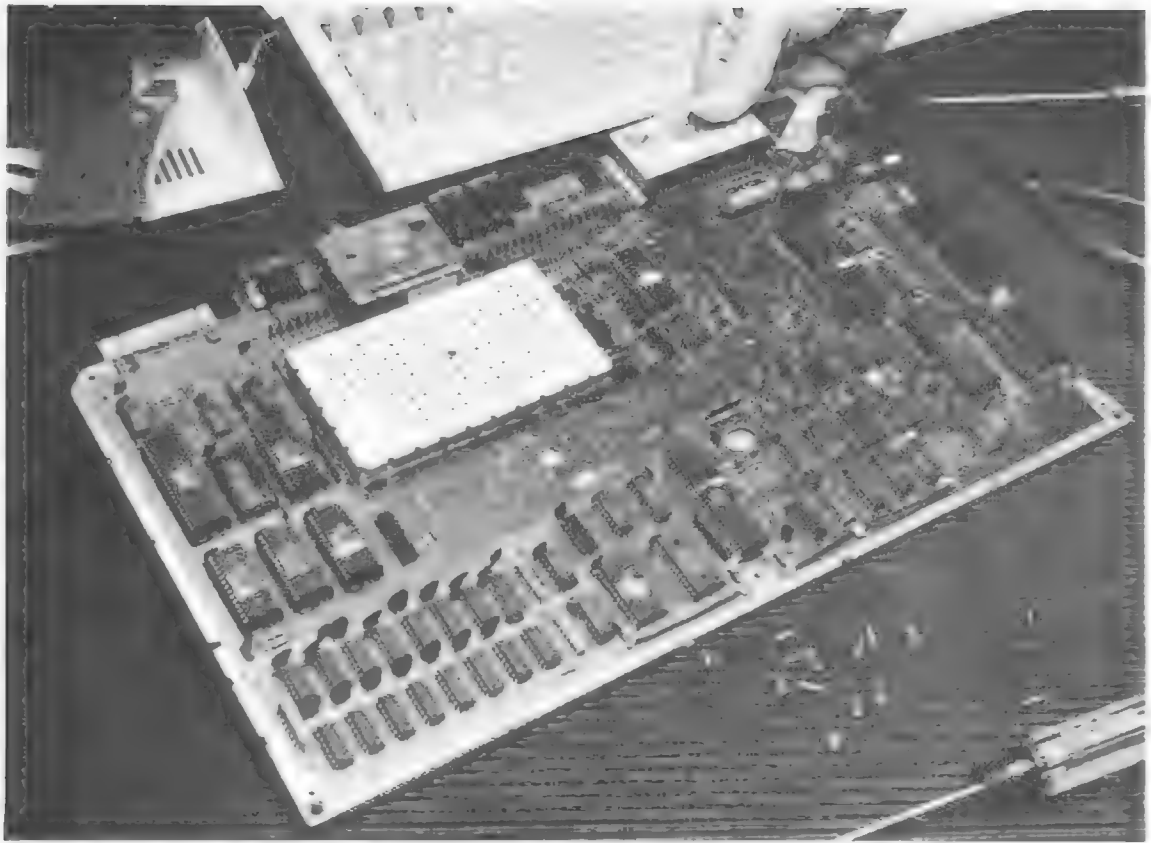


Fig. 2-12 The top shield is shown removed from the printboard. Note the white spots on some of the larger chips. The tabs on the top shield are dressed with silicone and touch these chips. This is a heatsink. The silicone transfers heat from the chips to the shield. During reassembly it is good practice to apply more silicone to these surfaces.

ways be bone dry. The idea of dusting and cleaning is not to make the board shiny and spotless. All you want is good air circulation and a clear view of the circuits on the top and bottom of the printboard.

A good preventative item to use is a dust cover for the C128. If you place it on the C128 religiously after you use it you will avoid most of the dust problems that occur. The only precaution needed with a dust cover is: do not put the cover on while the C128 is still warm from operation. Let it cool down first. The dust cover traps the heat somewhat and might cause difficulty.

STATIC ELECTRICITY

If you are going to be handling printboards and integrated circuits, then you should be briefed on

their worst enemy: static electricity. A silicon chip is considered sturdy and reliable in normal operation mounted on a printboard, but it is in serious jeopardy when it is loose and not in the protective board environment.

For example, if you walk across a carpeted room on a dry day, and reach for a loose chip, and a static spark flashes from your finger to the chip, then odds are that the chip has just been electrocuted. Oftentimes, you cannot stop the static buildup of the electric charge on your body. How can you avoid this problem? There are ways. Let's examine the situation first before discussing the preventative measures you can take.

Two general types of chips exist that you'll find in the C128. One is called a TTL, which stands for

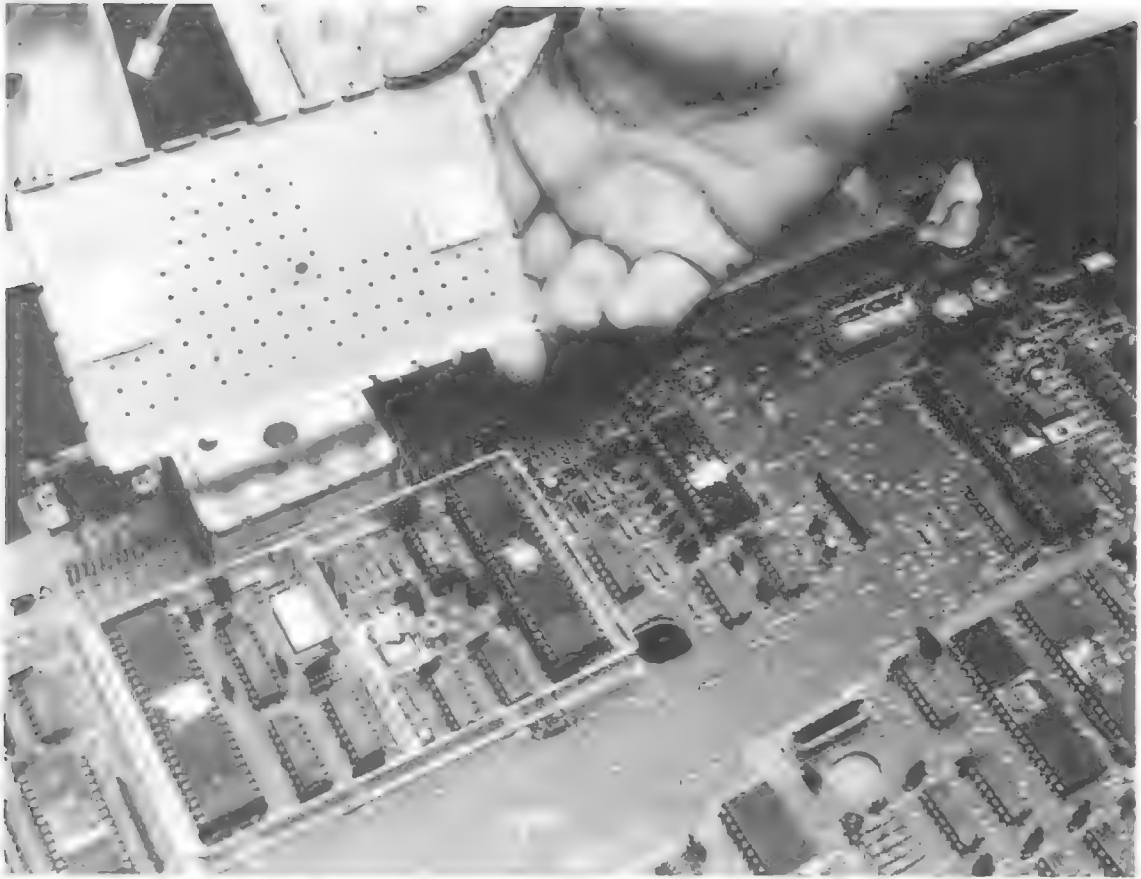


Fig 2-13 In the video box are eight more chips. Two of them have silicone paste and touch a heatsink tab on the box cover

Transistor-Transistor-Logic. The other is called an MOS, which stands for Metal-Oxide-Semiconductor. More detail on the construction and operation of these chips is in Chapter 4.

Among the chip's vital characteristics is one called Threshold Voltage for Electrostatic Damage. This describes the amount of static electricity the chip can withstand without being destroyed. The average TTI is not able to take a jolt of 300 volts or more. An MOS can't stay alive if a static shot of 250 volts or more is applied. Did you ever wonder how much voltage there is in a spark that leaves your body for a doorknob on a dry day? It could easily be 3000 volts! As you can see, it is an easy matter to lose a chip if you do not take proper precautions.

What is Static Electricity?

Static electricity involves the buildup of electric charges on the surface of an insulator. The charges are called electrons. The insulator will get charged if there is an excess or deficiency of electrons. An insulator with an excess of electrons is said to have a negative charge. When there is a deficiency of electrons, then a positive charge is present. The voltage developed gets larger as the charge increases. It doesn't matter which type of charge is on the insulator; either one will deal the chip a death blow.

Static electricity results from friction between different types of materials. Your-shoes-on-the-carpet is one way to produce this friction. As you walk around the room, you charge up. The rug and your shoes are both insulators. The charge is built

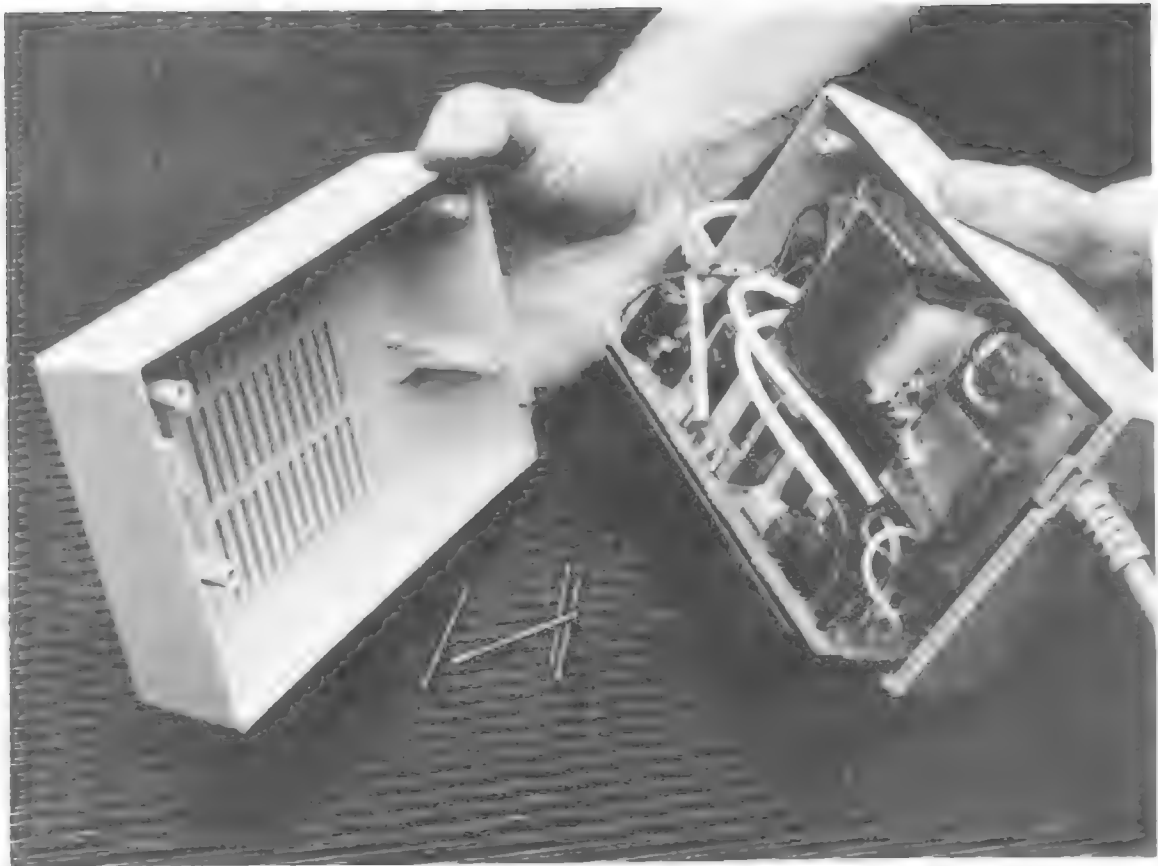


Fig. 2-14. Some of the C128 power box units have these four bolts holding the case together

on your shoes. The charge is then transferred to your body, which is a conductor.

The charge on an insulator remains at the spot, where it develops. The charge on a conductor, however, spreads itself evenly throughout the conductor. That is how we can get rid of the charge before it can kill chips. If we, as conductors, touch an earth ground, then the charge quickly leaks from us to the ground. The only problem there is: only the charge that was in our body is gone. The charge that is still on our shoes or on our clothing remains. Therefore, correct-chip-handling must still be observed even if we are grounded properly.

To help alleviate the problem, it is a good idea to wear conductive clothing. For instance, leather soled shoes are better than rubber soled shoes, and cotton clothing is more conductive than nylon.

Wrist Strapping

One technique that professional servicers use to deal with static electricity is called "wrist strapping." The wrist strap kit is available in electronic supply houses. RCA puts one out called an Antistatic Kit. It consists of a static dissipative mat, a light weight wrist strap, a coil cord, and a six foot grounding cable.

In series with the wrist strap, cord and ground is a resistor, typically about a megohm. Figure 2-18 shows the grounding scheme and Fig. 2-19 is an illustration of the RCA kit hookup. Any charge that builds up in your body will immediately be shunted off to the ground with this arrangement. Note that this is an electrical earth ground. An electrician usually attaches his ground to a cold water pipe for a good earth ground. If you are in doubt about the

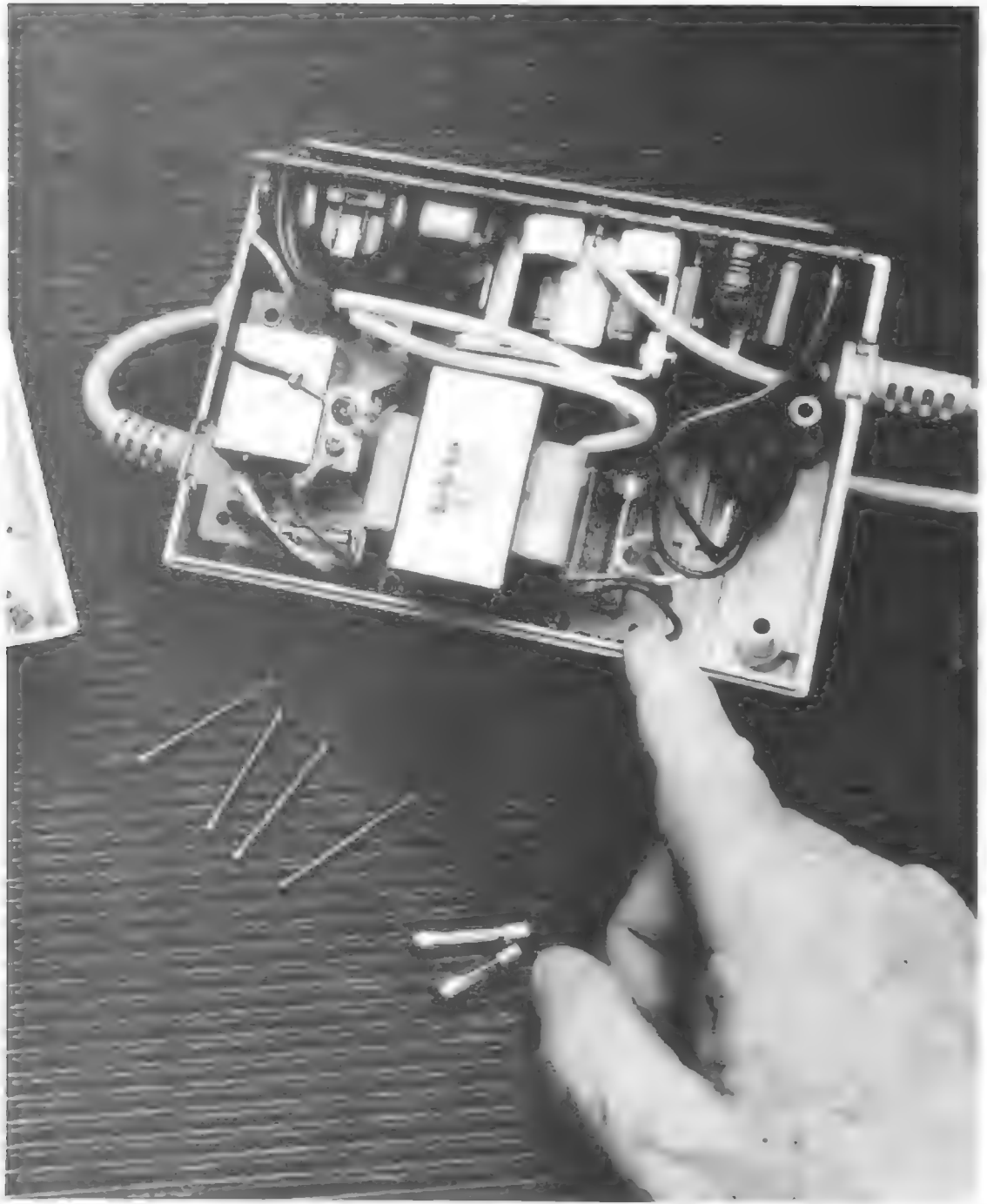


Fig. 2-15. Inside the power box are some small circuit boards and two easy to replace fuses

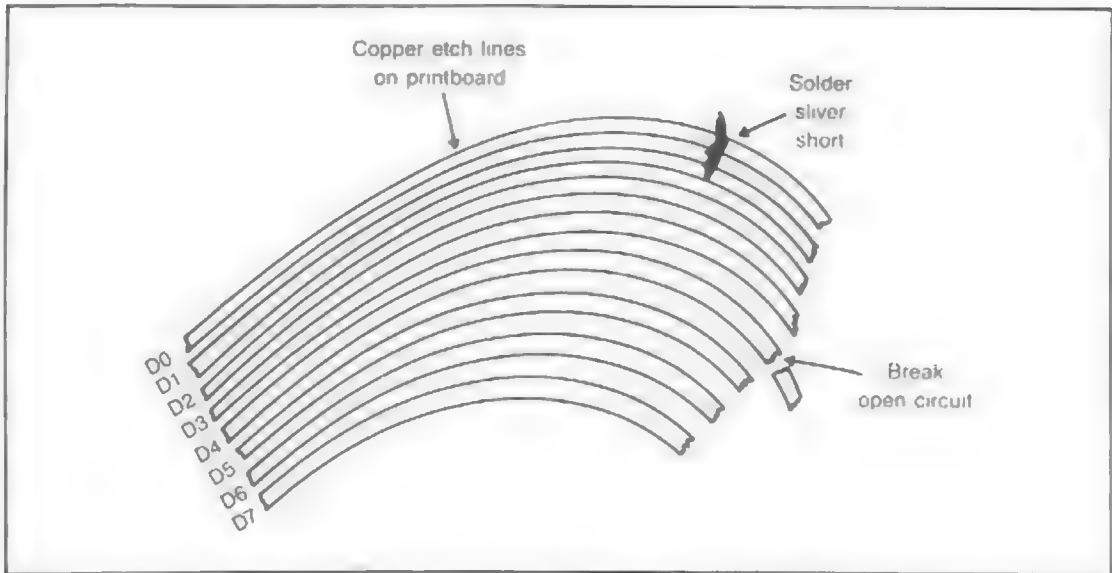


Fig. 2-16. If a sliver of solder should fall across wiring strips it will cause a short circuit. Should a copper etch track break it could be an open circuit.

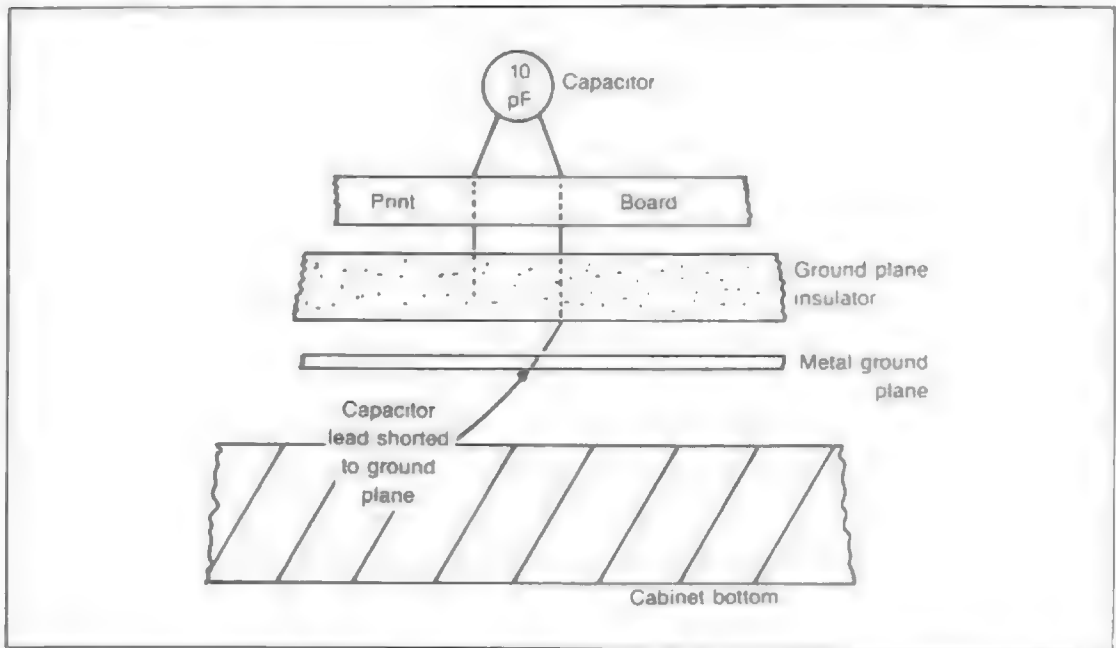


Fig. 2-17. The printboard is the top of a sandwich. Beneath the board is an insulator. Under the insulator is the metal ground plane. If a component lead should pierce the insulator and contact the sandwich bottom, the ground plane, then a short circuit could develop.

Table 2-1. This is a list of chips that can be quick-tested with a hot-warm-cool feel test. If the condition of the chip does not match the chart, then the chip could be defective.

Chip	Operating Condition
U1 CIA1 6526	Cool
U4 CIA2 6526	Cool
U6 MPU 8502	Warm
U10 MPU 280	Warm
U11 PLA 8721	Warm
U7 MMU 8722	Warm
U5 SID 6581	Warm
U22 VIC 8563	Hot
U21 VIC 8564	Hot
U32 ROM1	Warm
U33 ROM2	Warm
U34 ROM3	Warm
U35 ROM4	Warm
U38-U53 Set of RAM	Cool
U18 Character ROM	Warm
U19 Color RAM	Cool

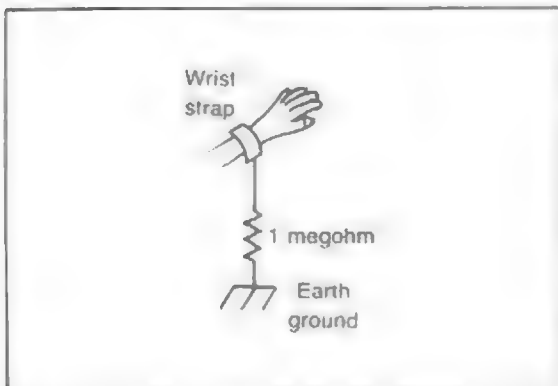


Fig. 2-18. In order to keep static electricity on your body at a harmless, low level, commercially available wrist strap systems can be used. They provide an escape path to earth ground for the unwanted charges.

earth ground, then consult an electrician to be sure you are connected correctly. If you do not connect to a true earth ground then the connection could be useless and not discharge your static voltage.

It is also vital for the wrist strap to be used only while you are working with equipment that is completely disconnected from any sort of electric power!

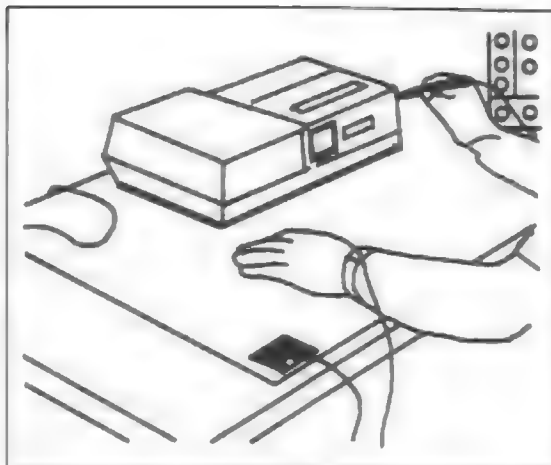


Fig. 2-19. RCA markets an antistatic kit to keep you and your workbench as static free as possible. It consists of a static charge dissipative mat, a lightweight wrist strap, coil cord and a 6-foot grounding cable that constantly drains static charges to ground. It is used **ONLY** while you are working with equipment that is completely disconnected from any sort of electric power!

While you are wearing the strap system, then you are securely connected to ground. If you contact the 120 Vac, you could get a nasty electric shock. That is why the megohm resistor is in series with the cord. It is supposed to limit the amount of current if you do manage to contact the 120 Vac. To be safe, only connect yourself to the wrist strap system during chip handling on disconnected equipment.

Put the wrist strap on after you pull all power plugs on the equipment. Then you can take the C128 apart. Keep the strap on during any chip handling. You are producing static charges with every move you make. After the chips are safely installed in the equipment you can remove the strap. Once the strap is off you then, and only then, you can plug the C128 into the power line.

Additional Precautions

The professional electronic technician often uses a lot of other static discharge techniques to avoid killing chips with accidental static charges. First of all, he pays attention to the workbench. Workbenches for electronics are usually made of insulat-

ing materials like pressed wood. As insulators, they can build up charges. You can't easily discharge the entire work surface because a ground wire will only discharge the point on an insulator that the wire touches. To solve this problem, the technician will ground the printboard he is working on. Ensuring that the board is not connected to power, the tech will connect a jumper wire, with a one megohm resistor in series, from the board to earth ground.

When you receive a new sensitive replacement chip, you could find it plugged into a black piece of what looks like insulation. It is not insulation: it is a conductive material. With all the pins plugged into it, all the pins are more or less shorted together, which is a safe condition. When you pull the chip out of the material and free the pins, then the danger begins. Don't remove the chip from the material till the moment of installation, and then only when you are prepared by being grounded properly.

There are special tools for chip handling. A chip

extractor has a hole in the top that can be connected to ground when necessary. A companion tool is the chip inserter. It can also be connected to ground easily through the pin on the top (See Chapter 4).

Sometimes static electricity is not a problem and you can let up somewhat on your chip handling precautions. If the humidity is relatively high, and there is no trace of static charge, TTLs can be handled without fear of damaging them. MOS chips, though, should always get extra care. They could be killed at any time.

I don't mean to scare anyone concerning these chip handling problems. It is possible and even likely that you could successfully replace the chips in your C128 without these grounding measures. However, if you do have to replace a chip in your computer and it takes weeks until you receive delivery on it, you will get rather frustrated should it blow out from a static spark, due to careless handling, before it is installed.

3. Chip Location Guide

The previous chapter showed you how to take the C128 apart and free the printboard. Once the board is out in the open, you will see a conglomeration of chips, resistors, capacitors, transistors, copper connecting etch tracks, metal boxes, ports, switches and other things. It is like flying over a city and trying to figure out what is where. This chapter maps out the C128 environment and identifies all the main components so that you can know where you are if you make any service moves.

Figure 3-1 is the map. It is the Chip Location Guide. It shows the physical relative position of all the chips, transistors, ports and switches, dc test points, a filter capacitor, a diode and a couple of adjustment spots. During servicing, the Location Guide is referred to continually. Many repairs can be consummated using only the Location Guide as service information.

SPECIFIC INFORMATION

As you look at Fig. 3-1, you'll find that the front of the printboard is on the right of the page and that

the left side represents the rear of the board. Across the rear panel are seven ports and along the right side are four more, plus the off-on switch. Table 3-1 lists these components. The ports are called CNs and the switch, SW1. All the ports and the switch have connection points. They are all numbered. In Chapter 23 the voltages and logic states as well as the pin numbers are provided. The Location Guide zeros you in on the physical positions of the ports and switch so that you do not accidentally try to test the wrong port.

You can see that in front of the RF Modulator box there is another metal box on the printboard. The box has a metal top with some ventilation holes in it. The Location Guide shows this box uncovered and displays the interior. There are eight chips, three transistors and an important adjustment coil in the box.

The transistors are given a prefix Q. There are six transistors, Q1 through Q6, on the printboard. There are some more transistors in the power supply box and in the RF Modulator box. However, they

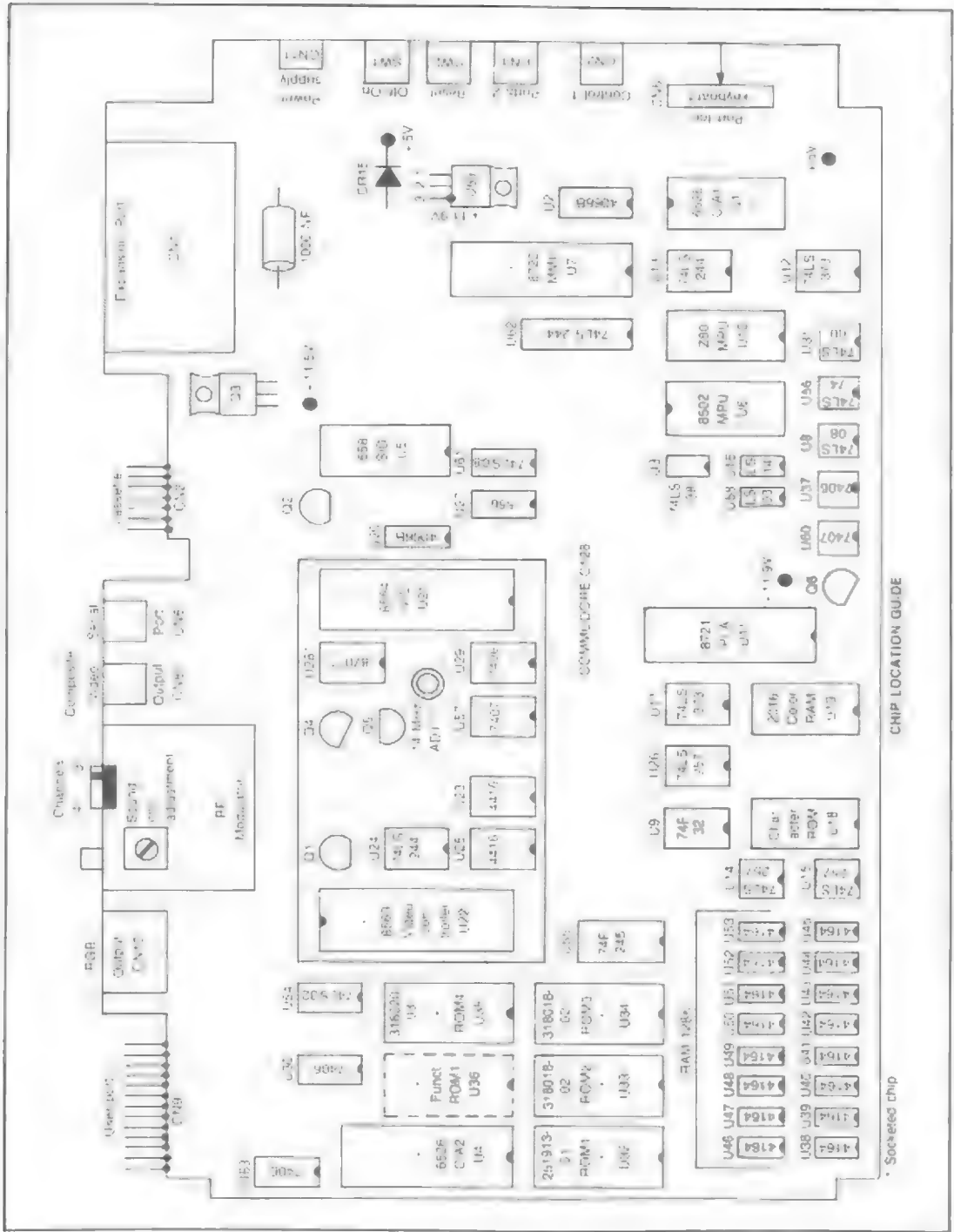


Fig. 3-1 The Chip Location Guide is the most used piece of service information during repairs

Table 3-1. This is a list of the connectors and switches around the top and right side of the printboard counting clockwise.

Rear and Side Panel Clockwise	Item #
User Port	CN9
RGBI Output	CN10
RF Modulator Box	M1
Composite Video Output	CN8
Serial Port	CN6
Cassette Port	CN2
Expansion Port	CN1
Power Supply Input	CN11
Off-On Switch	SW1
Reset Switch	SW2
Control Port 2	CN4
Control Port 1	CN3
Keyboard Port	CN5
Internal	
Serial Bus connections	CN7
Power Supply connections	CN12

Table 3-2. Six discrete transistors are on the printboard located as shown.

Transistor	Located at
Q1 2SC1815	Near top of U22, 8563
Q2 2SC1815	Near top of U5, SID
Q3 2SD880	Left of Expansion Port
Q4 2SC1815	Left of U21, VIC
Q5 2SC1815	Below Q4
Q6 2SC1815	Near bottom of U11, PLA

Table 3-3. The three power supply output voltages can be tested at these locations.

DC Test Points	Located at
+ 5.0 Volts	Bottom left of board at bottom of resistor R38
+ 11.5 Volts	Top right of board below the Base connection of Q3, on the left connection of resistor R3
+ 11.9 Volts	Bottom center of board next to pin 7 of U11, the PLA, at top connection of resistor R45

are covered separately in their own chapters. The six printboard transistors are listed in Table 3-2, along with reference locations.

On the right top quadrant of the Location Guide, a 1000 μ F filter capacitor and a diode CR15 are shown. They are important test points, as described in Chapter 24.

There are three dc voltage test points shown on the guide. They are listed in Table 3-3. They are also described in Chapter 24.

Then there are the 63 chips. They have U numbers—U1 through U63. They are scattered all over the board. They are different sizes, and vary from U59 (a 12 volt voltage regulator, with only three pins) to U22 (the Video Controller with 48 pins). The pin sizes can be judged on the Location Guide by comparison. Note that the largest chips are 48-pin, the next largest, like U6 the MPU are 40-pin, the ROMs are 28-pin, and so on. For verification of the comparative sizes, you can refer to the various Test Point Charts of specific chips in their respective chapters. Exact details in schematic form can be found in the Master Schematic in the back of the book.

Besides providing the U number for each chip, there is additional servicing information shown in the chips. First of all, the generic number of each chip is shown. With the generic number, you can purchase a replacement chip. Simply take the number to a supply house and give the counterperson the number. He is able to look up the chip and cross-reference the generic number over to the manufacturer's number of the brand that he carries.

Also, shown in each chip on the location guide is its given name. Table 3-4 lists all 63 chips on the board giving U numbers, generic number and given names. Lastly, in each chip is shown the keyway to identify the pin numbers.

The unfortunate thing about chips is: they can be put in upside down. The keyway won't prevent you from installing it incorrectly. However, the keyway does show you where the first and last pin numbers are. The keyway is a notch on the physical chip. It is shown as the black dot on the top or bottom of the chip. Pin 1 is to the left of the chip when the notch is at the top, as shown on U6 8502. The pins

Table 3-4. The C128 has room for 63 chips with U numbers.

U Number	Generic Number	Given Name
U1	6526	Complex Interface Adapter
U2	4066B	Quad Bilateral Switch
U3	74LS138	1-of-8 Decoder
U4	6526	Complex Interface Adapter
U5	6581	Sound Interface chip
U6	8502	Microprocessor
U7	8722	Memory Management Unit
U8	74LS08	Quad 2-Input AND Gate
U9	74F32	Quad 2-Input OR Gate
U10	Z80	Microprocessor
U11	8721	Programmed Logic Array
U12	74LS373	Octal 3-State D Latch
U13	74LS244	Octal 3-State Driver
U14	74LS257	Quad 2-Input Multiplexer
U15	74LS257	Quad 2-Input Multiplexer
U16	74LS14	Hex Schmitt Trigger
U17	74LS373	Octal 3-State D Latch
U18	390059-01	Character ROM
U19	2016	Color RAM
U20	4066B	Quad Bilateral Switch
U21	8564	Video Interface Chip
U22	8563	Video Controller
U23	4416	DRAM
U24	74LS244	Octal 3-State Driver
U25	4416	DRAM
U26	74LS257	Quad 2-Input Multiplexer
U27	556	Timer
U28	8701	Clock
U29	7406	Hex Inverter Buffer
U30	7406	Hex Inverter Buffer
U31	74LS00	Quad 2-Input NAND Gate
U32	251913-01	Read Only Memory
U33	318018-02	Read Only Memory
U34	318019-02	Read Only Memory
U35	318020-03	Read Only Memory
U36	Empty Socket for Functional ROM	
U37	7406	Hex Inverter Buffer
U38-U53	4164	DRAM
U54	74LS32	Quad 2-Input OR Gate
U55	74F245	Transceiver
U56	74LS74	Dual D Flip-Flop
U57	7407	Hex Buffer
U58	74LS03	Quad 2-Input NAND Gate
U59	7812	12 Volt Regulator
U60	7407	Hex Buffer
U61	74LS08	Quad 2-Input AND Gate
U62	74LS244	Octal 3-State Driver
U63	7406	Hex Inverter Buffer

are then counted counterclockwise down to pin 20, across the bottom to pin 21, and then up to the last pin 40.

When the keyway is mounted at the bottom of the chip, as U6 the Z80 MPU is on the Location

Guide, then pin 1 is directly to the right of the notch. The pins are still counted counterclockwise from pin 1. Simply count upwards to the top right pin 20, across to the left to pin 21 then down to the bottom left pin which is pin 40. The rule is that pin 1 and

the last pin number are always across from each other at the keyway, and that the count is counterclockwise.

All the chips on the C128 board, except for U59, the regulator, are called DIPs. DIP stands for Dual Inline Package. This refers to the fact that there are two lines of pins on the chip.

The Location Guide Perspective

The Landmark sketch in Chapter 1, Fig. 1-11, and this Chip Location Guide, Fig. 3-1, show the actual physical locations of the most important components on the board. They do not show the small support components or the circuit connections between the chips. You do not need to bother with the support components or the connections in the first stages of a repair.

These guides are designed to enable you to quickly find a suspect chip or transistor. You can remove and install the twelve chips in sockets. It shows you where the adjustments are. It lets you know where convenient test points are for power supply voltage readings. It keeps the ports straight. It lets you make all your visual tests quickly and accurately. It turns out that, with only the aid of the Location Guide, you will be able to take care of a large percentage of repairs, cleaning and adjusting.

When the simple measures do not produce a fix, then you must use more complex service information. First of all, you can test chips for logic states with the Test Point Charts. The charts are close up views with the logic states that should be present on all pins. The charts are also physical replicas of the actual chips. The Location Guide shows you where each chip is on the board.

For even more serious servicing, you will require the Master Schematic. It uses electronic symbols to represent the physical chips, transistors, capacitors, etc. You must, in your mind, be able to relate the symbols to the actual components on the board. The symbols bear no resemblance to the components. Also, the pin connections on the schematic are drawn for the convenience of the draftsman, and there is no rule such as counting counterclockwise. The pin connections can appear

anywhere on the drawn symbol. You'll pick up tips on reading schematics as you proceed through the book. Anyway, you'll find that the familiarity you gain with the Location Guide helps bridge the step of finding a part or connection on the board after reading the schematic.

CHIP SURVEY

Getting back to Fig. 3-1, the chips appear to be scattered helter skelter all over the board. Yet there is good design in the layout. The chips are wired up in groups according to the various jobs required in a computer. Let's check out the major chips on the board and take an overview of their operation.

Microprocessors

If we view the board in quadrants, from the front, the MPUs are in the lower right. They are the 8502 and the Z80 sitting next to each other. The 8502 is used to operate the C128 and C64 modes. The Z80 works when the CP/M modes are in operation. The 8502 is an upgrade of the old 6502 and the Z80 is the same one used for years. They do not operate together. Either one is working, or the other one is, according to the mode you are using. Chapters 12 and 13 cover the two processors in detail.

The MPUs have the job of addressing all the residents of the memory map around the board. They send and retrieve data from the memory, and process the data with the aid of internal registers, its arithmetic and logic center, and its control center. The MPU uses its internal registers as temporary memory locations to store results in progress, incoming instructions, and memory map addresses that will have to be contacted during the data processing.

The arithmetic and logic center does the data processing. It is known as the ALU for Arithmetic Logic Unit. It performs all the calculations. It also selects, sorts, and compares the information according to the instructions it receives. The control section is a traffic cop in the MPU that keeps the data traffic moving in the correct direction at the proper time.

Complex Interface Adapters

The two main I/O chips are the two 6526 Complex Interface Adapters referred to as the CIAs. U1, known as CIA1, is located in the lower right quadrant also, next to the two MPUs on their right. Directly to the right of CIA1 is the keyboard port. The keyboard port is wired directly to CIA1. If you look closely at the bottom of the actual printboard, then you can see the copper etch tracks connecting the plug to the chip. It is obvious then that CIA1 is the I/O port for the keyboard. In fact, that is the main job CIA1 performs.

However, CIA1, in its spare moments, also takes care of the I/O needs of the two nearby Control Ports, 1 and 2.

CIA2, the other 6526 I/O chip, is found in the upper left quadrant of the board. It is the chip that handles peripheral signals from its nearby ports. They are the User Port and the Serial Port.

The CIAs are 40-pin chips and do a lot of work in the C128. They are covered in detail in Chapter 19. Besides being an entrance and exit way for the above mentioned ports, they also perform a lot of other jobs: they possess shift registers and precise timing circuits; they have a time-of-day clock; they are also the source of many interrupts that permit the MPUs to service various circuits.

The Video Interface Chips

As mentioned earlier, the C128 operates as a C128, a C64 and as a CP/M program runner and writer tool. In the C128 mode and the CP/M mode, the machine is able to output both 40-column and 80-column displays. There are two video output chips—one handles the 40-column output and the other the 80-column output.

Both of the chips are found on the Location Guide in the upper left quadrant. They are under the metal shield of the video container below the RF Modulator box. On the right side of the video container is the 48-pin 8564 VIC chip. It is an upgrade of the 6567 VIC chip that is in old C64 machines. This newer VIC takes care of all the 40-column display needs for the C128 and CP/M modes. It also

takes care of the 40-column display that the C64 mode puts out.

On the left side of the metal box is the 8563 Video Controller chip. Its job is to produce all the 80-column displays that the computer is capable of. It produces 80 columns for the C128 and CP/M modes. The C64 mode can't use an 80-column display, so the chip is not used when the C64 mode is in operation.

The video output chips are the interface between the digital circuits and the analog video output circuits. The chips receive inputs from the digital computer circuits. Inside the chips, the digital signals are transformed into analog outputs. The analog outputs are various TV signals and RGBI signals. The 8564 VIC chip produces two TV signals, one called Luminance/Sync and the other Chroma. In the RF Modulator these signals are combined to help produce the Composite TV Signal. The two signals are also output individually to the Composite Video Output Port.

The 8563 Video Controller Chip receives digital signals also. It then converts the digital to analog Red, Green, Blue and Intensity signals. The signals are then output in the 80-column format. The 8563 also generates the required sync signals.

The video chips are the source of many forms of outputs. They produce variations of alphanumeric, semigraphics and pure graphics. One of the features of these machines is to be able to generate "sprites." A sprite is a high resolution programmable figure that can be put into a graphic display. It is used in sophisticated graphic programs. A sprite can be formed in all types of conceivable shapes and made to cavort freely around the display.

These complex video output chips can act somewhat like MPUs. They have addressing and programmable capabilities. In some cases it is possible to have the actual MPUs turned off and let a video output chip take over the computer during video processing. When a video chip is in charge, it conducts the addressing and control functions that the MPU normally performs. Details on the 8564 chip is found in Chapter 20, and the 8563 is covered in Chapter 21.

The Sound Interface Device

In the upper right quadrant, just right of the video box is the 6581 SID chip. SID is the companion of the video chips that produce the sound to go with the TV display. SID is a 28-pin chip and is a programmable resident of the memory map. On the printboard it is known as U5.

SID is wired to the data bus, and also to enough lines of the address bus so that it can be addressed. The details are supplied in Chapter 22. SID has direct inputs from the MPUs and can be programmed easily. SID's output is audio and goes to an audio output amplifier and then to the RF Modulator. From the modulator, the audio goes to the RF Output plug. SID also sends audio directly to the Composite Video output port. Sound can then be heard from a home TV acting as a monitor or from an actual monitor with sound capabilities.

SID is the same chip that is also found in the C64 machines. It is referred to as a three voice electronic music synthesizer. It is used to generate interesting and exciting sound effects to go with the graphics created by means of sprites and other video creations. SID has a wide range of frequencies that it can produce. There is a high resolution control of the frequencies, harmonics, and volume.

SID is able to generate four different types of audio waveforms. They are known as triangular, sawtooth, rectangular and white noise. The frequency of each waveform can be varied individually. Once generated, the sound can be fed to an envelope shaper circuit. The audio arrangement of sustain level, attack, decay and release rates is made. With these variables you can produce audio that imitates musical instruments and other noises. Chapter 22 tells you how to test SID for these sound effects.

SID, like the video chips, is also a digital-to-analog converter. The digital inputs from the programming are changed to analog audio outputs.

The MMU and the PLA

Once the address bits leave the MPU, then they are processed in a very complex way. U7, the 8722

Memory Management Unit, in the upper right quadrant, and U11, the 8721 Programmed Logic Array, on the border between the bottom two quadrants, do most of the processing. Chapter 15 covers the 48-pin MMU. Chapter 14 goes over the operation of the other 48-pin PLA chip.

In general, the MMU uses its address and data bus inputs and outputs to access and handle the two banks of 64K RAM in a custom styled manner. There are a number of ways that the RAM could be used. The style of use is determined by a lot of things, most important of which is the mode you want the C128 to operate in. You could say that the MMU controls the operating mode.

The PLA works as an assistant to the MMU. The PLA also receives an input of address bits, but in addition it receives inputs from the MMU. The PLA in turn generates a lot of chip selection signals. While the MMU is addressing the RAM banks in the memory map, the PLA selects the other chips. It selects the ROM to be used—if a cartridge is plugged into the C128 it will select it. It is able to select the color RAM chip, the VIC registers, the Character ROM and all the I/O devices.

Between the MMU and the PLA, most of the chip selecting and location addressing is conducted. The operation details are covered in Chapters 14 and 15.

The 128K RAM

The C128 is blessed with 128K of RAM. The RAM is contained in 16, Dynamic Random Access Memory 4164 chips. The MPUs in the C128 are the so called eight-bit types. An eight-bit MPU can directly address only 64K. Therefore the 128K is broken up into two 64K banks, called Bank 0 and Bank 1. With the help of the MMU chip, the processor in use is then able to address each bank in turn. The full 128K of RAM is located in the bottom left-hand corner of the printboard.

The 16 chips in the two banks are all identical 16-pin chips. These RAM chips are the storehouse for the MPU. The chips are able to store data and then transfer the stored data back to the MPU when it is addressed. Chapter 6 goes into the details.

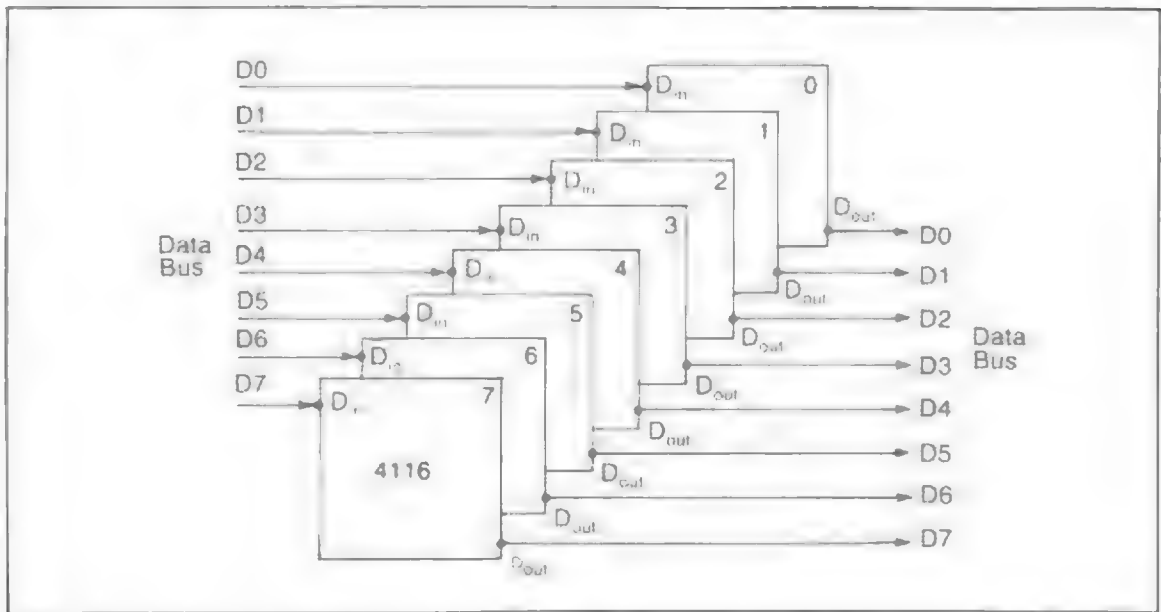


Fig. 3-2 Dynamic RAM is organized in bits. An addressed location is found split up on eight chips. Each chip outputs one bit of the byte at the location.

In one bank there is 64K of RAM. In each of the 64K locations there is one byte of storage space. There are eight bits in each byte. Each bit is connected to one of the eight corresponding lines of the data bus, D7 through D0. The bits in each location are numbered 7, 6, 5, 4, 3, 2, 1 and 0, as shown in Fig. 3-2.

In this type of dynamic memory, you'll find that a location of eight bits is not on a single chip but is spread over the eight chips. Each chip of the eight in a bank contains one bit of a location. In order to address a location, you must contact all eight bits. Chip number 7 of the bank set contains all the number 7 bits for all of the 64K locations. Chip number 6 has all number 6 bits and so on. When the MPU addresses these RAM locations, then it is actually addressing all eight chips at the same time. It gets one bit of the location from each chip.

Dynamic RAM, such as these 4164 chips, stores bits as charges, or noncharges, in a grid of tiny capacitances. The charge in the capacitance is very fragile and can only exist for a few milliseconds before it leaks off. It must be recharged every few mil-

liseconds in order to hold the charge. This recharging, called "refreshing," is conducted by a special refresh circuit in the VIC chip. The refreshing is conducted continually by VIC as long as the C128 is running.

The ROMs

Directly above the 128K RAM are located the ROMs one, two, three, four and an empty socket for a possible fifth special ROM. The fifth ROM is called "functional ROM 1." The five ROMs are U32, 33, 34, 35 and 36. They are all 28-pin chips. They are covered in Chapter 7.

The ROM is unlike the RAM in that it is a Read Only Memory. The ROM is already filled with permanent bits that are "burnt" into the individual locations. Also, unlike the dynamic RAM memory, a location is not spread over a group of ROM chips. Each addressed location is found complete on a chip. All eight bits of a byte location are together on one chip, as Fig. 3-3 indicates.

The main difference between a ROM and a RAM is: the ROM can only be read by the MPU.

Memory map addresses	Bit positions							
	7	6	5	4	3	2	1	0
53255								
53256	L	L	L	H	H	L	L	L
53257	L	L	H	H	H	H	L	L
53258	L	H	H	L	L	H	H	L
53259	L	H	H	H	H	H	H	L
53260	L	H	H	L	L	H	H	L
53261	L	H	H	L	L	H	H	L
53262	L	H	H	L	L	H	H	L
53263	L	H	H	L	L	H	H	L
53264								

Fig. 3-3. This ROM is organized in bytes. Each addressed location is on one chip and contains eight bits.

It cannot be written to successfully. Writing to a ROM is useless. It does not respond.

The permanent programs contained in ROM chips are the controlling brains of the computer system. The MPU is just a talented workhorse taking its orders from the ROM system operating programs. The MPU has no mind of its own. That's why, if a ROM chip becomes defective, the MPU starts running without direction and fills the display with garbage.

Inside the ROMs

The ROMs are located on the left side of the board, in a clump, about the middle of the board. There are five ROM locations; four are filled with

chips while an empty socket is present for use with a fifth ROM chip. The four ROMs in operation all have 16K locations. The fifth ROM, when it is used, has a 32K socket awaiting it.

ROM1, U32, is the closest ROM to the edge of the board. It is the controller when the computer is operating in the C64 mode. The PLA selects ROM1 during C64 operation and leaves the rest of the ROMs turned off. ROM1 contains the C64 operating system. This is a combination of BASIC 2.2, the C64 Kernel and other operating forces like the 40 Column Editor. There is more detail on these systems and the contents of the rest of the ROMs in Chapter 7.

ROM2, U33, is next to ROM1, directly to the right. It is selected by the PLA when the computer

is put into the C128 mode. It operates the BASIC I.O Version 7.0.

ROM3, U34, immediately to the right of ROM2, also operates in the C128 mode and controls the BASIC HI Version 7.0. In addition, ROM3 contains the operating system for the Monitor that you can use when you enter machine language programs from the keyboard.

ROM4, U35, is found directly above U34. It is also needed for C128 operation. It contains the C128 Kernel. Kernels in these Commodore machines take over and control all the input, output and memory management functions of the computer. Most of the time the work consists of transferring data from the MPU to the memory and back again. Another vital job is to verify the data that travels from place to place. The operating system has special load, store, and verify routines that BASIC calls upon to conduct its business.

The Kernel is a special operating system adjunct that attempts to keep the operating system up to date as time goes by. These C64 and C128 programs will, in the future, be improved and upgraded. The Kernel is therefore built with special jump tables that are designed to accommodate changes. That way, the machine language routines that you might write will not become obsolete and will work with newer versions of the C64 and C128. The Kernel attempts to maintain the compatibility of your present machine with future machines. There is more about the Kernel operation in Chapter 7.

The empty socket to the left of ROM4 is called U36 and is inoperative till a special functional ROM is plugged into it. Any such ROM will have special applications. However, there is no sense in worrying about it if it's not there. You can concern yourself with the socket since it is wired onto the board and will have some voltage and logic states on the pins. These are covered in Chapter 7.

Other Board Landmarks

There are more, not so prominent landmarks, all over the board that you will get to know if you spend time with the printboard—examining voltages, logic states and scope pictures. Most obvious are the RF Modulator box at top left, the expansion port at top right, the user port at top left, the off-on switch on the top right end of the board, the cassette port to the left of the expansion port, and so on. There are three dc voltage test points on the board. One is just below the expansion port to the left. It should read +11.5 volts dc when normal. A second test point that should read +5 volts dc is in the lower right-hand corner of the board. Thirdly, to the right of the PLA is a +11.9 Vdc point.

Other important spots on the board are: the places where the six transistors, Q1 through Q6, are found; the two adjustments to the sound coil in the RF Modulator and the 14 MHz to the left of VIC; the Character ROM U18; the Color RAM U19. All of these minor landmarks are discussed in detail in their respective chapters. They are all important test areas used during troubleshooting.

The chip location guide found in this chapter will be the most used piece of service information for the C128. It is a map of the printboard. It shows all the major components on the board and helps locate items.

It is easier to use than a photo of the board because it does not confuse the situation with all the wiring and hundreds of small components and connections. For these fine details you can refer to the board itself, the Test Point Charts, and the Master Schematic in the appendix.

The Location Guide provides you with clear locations of the 63 chips, the six transistors, the 10 external ports, two important adjustments, three dc test points and a few other items. Begin with this piece of service information.

4. Chip Changing Techniques

When chip trouble strikes your C128, then you follow these steps.

- (1) Interpret the symptoms and come to some sort of diagnosis.
- (2) Decide which chip or chips are the prime suspects.
- (3) Disassemble the machine to gain access to the suspects.
- (4) Run some tests on the involved chips.
- (5) If a chip is bad, then it must be replaced.

While the first four of the above steps require a steady hand, they are relatively danger free. Not so with step number 5. That's because inside the chips are transistors the size of germs. These transistors, possibly thousands upon thousands in a single chip, usually very rugged and reliable while the chip is connected in its designed circuit, become sensitive and fragile when they are free and out of their protective circuit.

The danger begins when a chip is either plucked out of its socket, or otherwise removed from the printboard. Some chips are more vulnerable than others. The dangers consist of physical pulling and pushing, heat, voltage and static electricity. As a general rule, the smaller chips are less sensitive and the larger chips are more sensitive. The larger the chip, the more care it requires when it undergoes movement to and from the printboard.

Even though some chips are more rugged than others, it is a good idea to treat them all with great care. This chapter covers the general techniques needed to handle the various forms of C128 chips when taken in and out of sockets, or desoldered and resoldered. When you use the correct techniques and tools, you will be able to change and test chips with the least amount of danger.

THE RUGGED CHIPS

Table 4-1 is a list of the more rugged chips in the C128. You'll notice that most of them start with

Table 4-1. The TTL rugged chip types are often 74—types.

U Number	Generic Number
U3	74LS138
U8	74LS08
U9	74F32
U12	74LS373
U13	74LS244
U14	74LS257
U15	74LS257
U16	74LS14
U17	74LS373
U24	74LS244
U26	74LS257
U27	556
U29	7406
U30	7406
U31	74LS00
U37	7406
U54	74LS32
U55	74F245
U56	74LS74
U57	7407
U58	74LS03
U59	7812
U60	7407
U61	74LS08
U62	74LS244
U63	7406

74. These are usually TTL chips. TTL stands for Transistor-Transistor-Logic. The TTL name is given because, as shown in Fig. 4-1, the input transistor on such a chip is a bipolar type but is endowed with double emitters.

A bipolar transistor (aside from the TTL), without going into a long dissertation, is one with three connections called the Emitter, Base and Collector: E, B and C. Figure 4-2 depicts the way typical npn and pnp bipolar transistors move electrons and holes that electrons can be kept in. The transistors are called bipolar because conduction takes place in two directions at the same time. The electrons, carry a negative charge travel in one direction between emitter and collector, while holes which carry a positive charge travel in the other direction. This type of conduction is different than Field Effect Transistors, FETs, which is discussed next. The FETs move either electrons or holes but not both at the same time. If you desire more details on bipolar transistor theory, please look it up in the many books available from TAB.

The TTL is a chip that evolved from earlier chips such as the RTL and DTL. The TTLs have

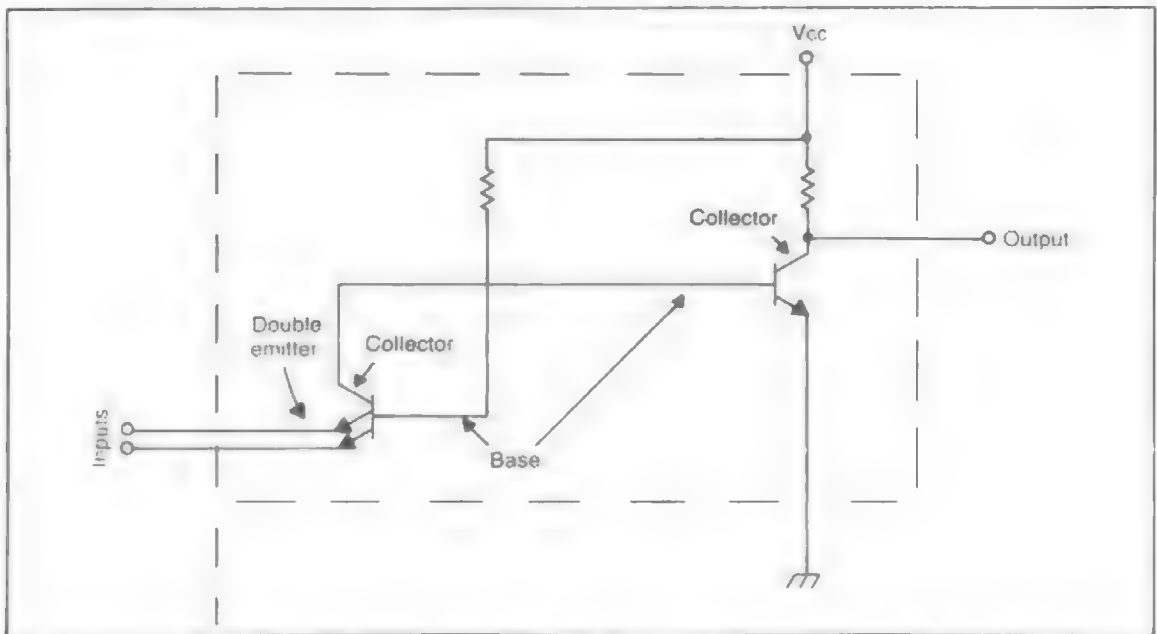


Fig 4-1 The TTL (Transistor-Transistor-Logic) chips are based around special transistors with double emitters.

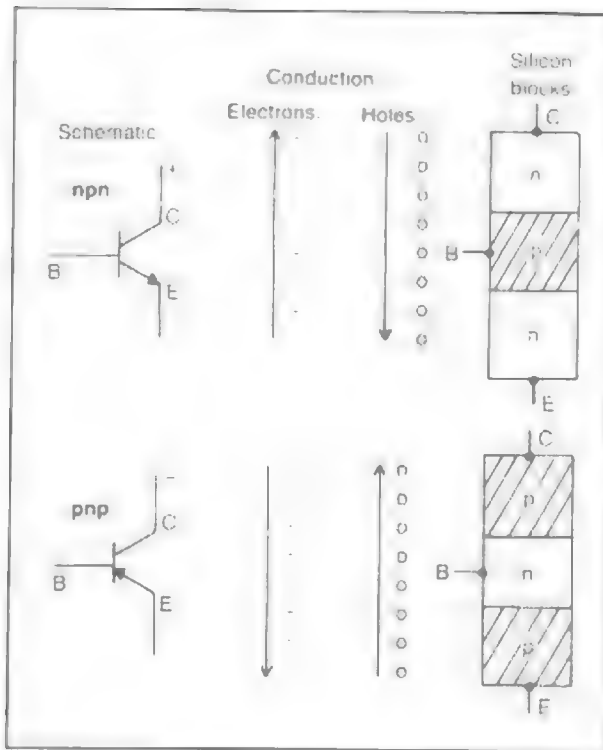


Fig. 4-2. Bipolar transistors are so called because they have electrons moving in one direction while holes are moving in the other.

combined a lot of the important features of these earlier chips. The RTL, which stands for Resistor-Transistor-Logic, was among the first forms of integrated circuits. It was an inexpensive chip that was easily wired up with larger discrete components such as resistors and capacitors. However, the RTL was highly susceptible to voltage noises, and it had a low fanout ability. *Fanout* is the characteristic a chip has to drive a number of parallel loads. The RTL can only drive a few loads.

The RTL gate in Fig. 4-3 uses two resistors in the base inputs of the two npn bipolar transistors. That is why it is called a Resistor-Transistor-Logic chip. This particular configuration is called a NOR gate. There will be more about NOR gates in Chapter 10.

The DTL gate in Fig. 4-4 uses diodes in the input circuit rather than transistors. This change of component makes the gate faster, gives better noise immunity protection due to diode clipping, and increases fanout characteristics. In addition the DTL

permits a large fan-in. *Fan-in* is the ability of the chip to accept parallel inputs. The diodes are able to isolate the gate input circuits from the preceding stages and many parallel inputs can be connected without loading the input. The basic DTL circuit is called a NAND gate. There is more about NAND gates in Chapter 10.

The TTLs in your C128 evolved from these two basic circuits. In 1961, Thompson invented the TTL. It is like a DTL in that there is a diode action in the input. The input is through the double emitters of the transistor. The extra emitters are pn junction, just like diodes. All the inputs to the chip can enter through the multiple emitters and be isolated in the same way the DTLs are isolated. The circuit operation is quite like the DTL.

There is a whole family of TTL chips—more than 200 of them. A few mail order companies are listed in Table 4-2. You can purchase replacement chips from them or from local electronic supply houses.

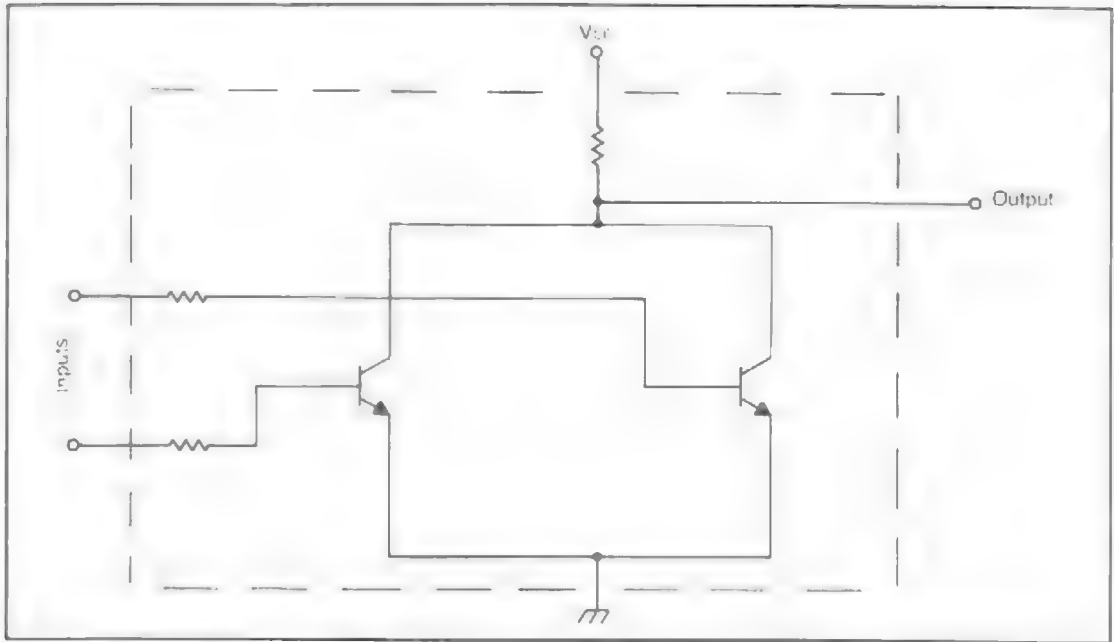


Fig. 4-3. The RTL (Resistor-Transistor-Logic) chips get their name from the resistors in their input.

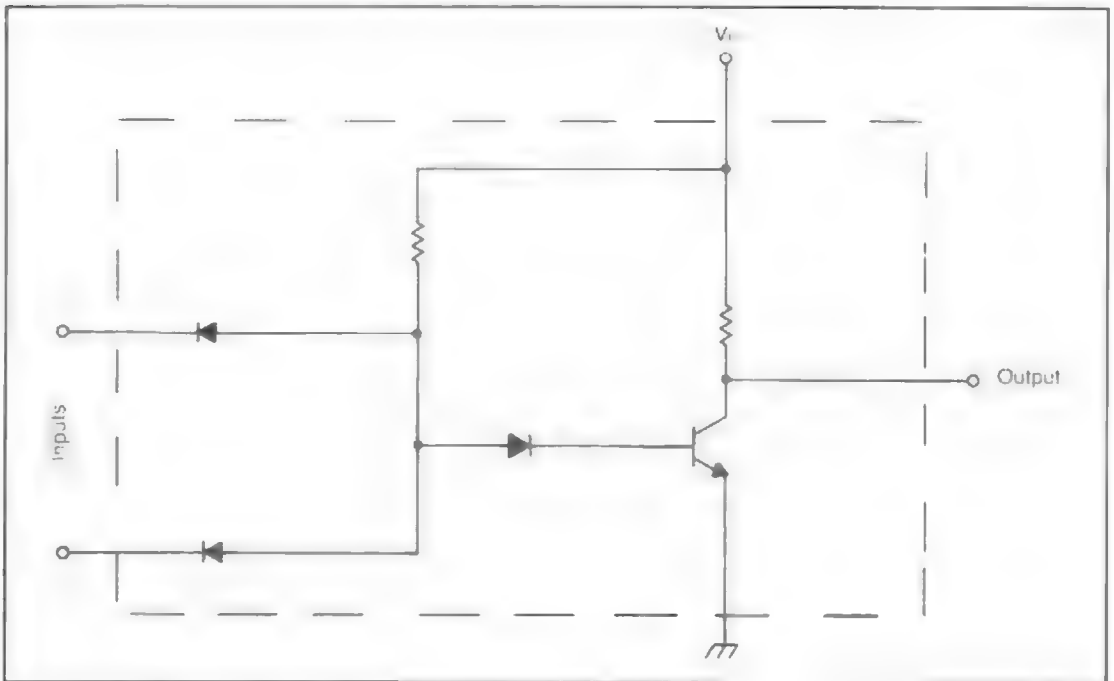


Fig. 4-4. The DTL (Diode-Transistor-Logic) chips have diodes in their input.

Table 4-2. List of Mail Order Parts Houses.

Mail Order
Jameco Electronics 1355 Shoreway Rd. Belmont, California 94002 Orders (415) 592-8097, Inquiries (415) 592-8121
JDR Microdevices 1224 S. Bascom Avenue San Jose, California 95128 Orders 800-538-5000, Inquiries (408) 995-5430
DIGI-KEY Corporation P.O. Box 677 Thief River Falls, MN 56701 1-800-344-4539

While the number 74 indicates the chip is a TTL, there are letters such as L and S that also have special meanings. The letter L stands for low power. When there is an L in the chip number, then it means that the chip uses 80% less power than a chip without the L designation. However, the lower power dissipation is at the expense of slower switching speed.

That is why there is usually an S accompanying the L in the nomenclature. The S stands for Schottky clamped diode. When the S is in the name, then there is a Schottky barrier diode clamp in the base circuit that speeds up the switching action. The S characteristic compensates for the slowdown caused by the low power characteristic. Therefore, if you are called upon to replace a 74LS type chip, use another 74LS type and not a plain 74 type. While the two types are functionally about the same, the exact replacement is always the best way to go. If you have no choice but to make a change, just be aware of it in case some other trouble symptom should suddenly appear. In some cases, the substituted chip will not work properly.

TRISTATING TTLS

The main business of the digital circuit in a computer is the processing of two logical states. The two states are known to a machine language programmer as 1 and 0. To a technician, the states

are called high (1) and low (0). The high and low refer to high and low voltages. Loosely speaking a high is +5 volts dc and a low is zero volts dc. These voltages speed through the digital circuits, and as they travel they are changed and changed again from high to low and back. That is all that digital circuits do. That is what the processing consists of: changing logic states.










At all the test points, which is every connection or pin you can lay a probe onto, you will be testing for logic states. As long as the called for logic state is present, then the pin is considered okay. Should a test point have the wrong logic state, then that could be a clue that could lead you to the trouble spot. Table 4-3 shows the two most important logic state testers: the logic probe and the vom. The high and low readings and their corresponding voltage levels are shown. Also shown in the bottom column is the third possible state a connection could be in.

This third state (tristate, three-state or floating, as it is called) can be confusing. It isn't actually a logic state, despite its description. It is a state of being, that the test point is in, when the computer is on but there is no definable output. It is the condition that the test point exhibits when the chip is energized but is turned off in the circuit. The TTL output, being shut off, assumes a high impedance condition. Any voltage level that might develop on the tristated test point is simply static noise buildup, not logic. In contrast, the zero voltage state is a connected state, the test point reads a voltage, but the voltage is zero.

When a chip is tristating, there is no definable output voltage. As a matter of fact, if you should take a dc voltage reading with a vom, there will be a voltage reading in the vicinity of 2 volts. It is a result of accumulated static electricity, not a logical high or low. Should you read a tristating test point with the logic probe though, then no LED lights will glow, indicating the tristate condition. There will be more about testing TTLS in Chapters 10 and 11.

Some TTLS are built with a tristating ability. They have a special input stage that is able to disable the TTL gate upon command. Figure 4-5 shows a chip with the disabling stage. There are three npn

Table 4-3. The vom and logic probe are the best pieces of test equipment with which to check chips.

LOGIC STATE	VOM READING	LOGIC PROBE LED LIGHTS
HIGH	2.3V to 5.0V	HIGH  LOW  PULSE 
LOW	0V to 0.8V	 HIGH  PULSE 
THREE-STATE FLOATING	0.9V to 2.2V	  

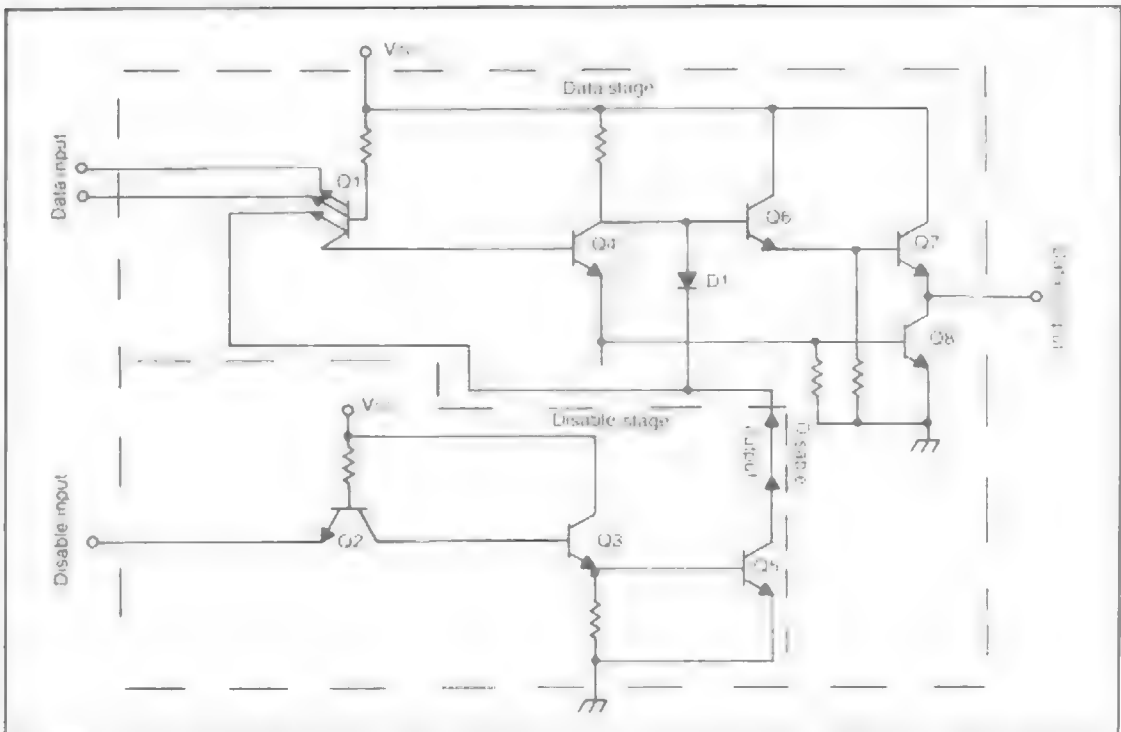


Fig. 4-5 The top part of this chip is a TTL NAND gate. The bottom is a disable stage. The disable stage can act as a switch to turn the NAND off and on. When off, the NAND is tristating or "floating."

transistors and a diode in the disable stage. They are Q2, Q3, Q5 and D1. When the disable input is a low, Q2 will turn on fully, and as it is called, "saturate." As a result of Q2 saturating, Q3 and Q5 will turn off. This turns off the output of the disable stage and the stage is disconnected from the rest of the chip. The rest of the chip operates as if the disable stage is not present. The normal processing of 1's and 0's continues unabated.

Should the input of the disable stage go high, Q3 and Q5 will then conduct and kill the output current at Q4. This makes the output transistors of the gate, Q7 and Q8 stop conducting. This makes the chip tristate. Should you measure the voltage at the output, you'll find neither a defined high nor low. The only voltage there will be undefined, somewhere between a high and a low. Technicians might say, "it's floating."

During bench troubleshooting, you can test the output of a three-state chip at its output. You can also produce a defined output by causing a high at the disable input to effect the logic condition or a low input to get the undefined performance.

The three-state effect can be used as a valuable servicing technique. Sometimes when a chip dies, then it produces a three-state condition at its output. At other times, a three-state condition is present during normal operation. You can use the reading you obtain, compared to what the Test Point Chart reads, to figure out whether the condition is normal or if it indicates trouble.

The TTL is easily tested with either the logic probe or the vom. The only difficulty is the tiny size of the chip feet. The feet are test points, and you must have a bright light and a magnifying glass. You must take care to touch only the foot being tested while the C128 is on. Avoid an accidental shorting of two test points. Most of the time you would probably be lucky if you shorted two points, but on occasion you'll burn something out by careless use of the probe.

The vom, measuring a TTL, will normally reveal a logical one, or high, by reading a voltage between 2.3 volts and 5 volts dc. The logical zero, or low, will display itself on a vom by reading a voltage between 0 and 0.8 volts dc. During a tristate condi-

tion, the vom will read somewhere in between, from 0.9 to 2.2 volts dc.

The logic probe has LED lights to denote the condition of a test point. Highs light the HIGH light, and lows make the LOW light shine. When the test point is tristating, none of the LEDs light at all. Table 4-3 rounds up all the high, low and three-state readings.

Most of the bench readings performed during servicing are either with the vom or the logic probe. The servicing consists of a search and seek expedition of examining tiny test points.

In the C128 there are a lot of TTL chips. They are covered in Chapter 8. They are, for the most part, the smaller support chips and are soldered onto the board without the benefit of having their own sockets in which to reside. The TTLs, as a rule, are fairly rugged and can take handling and soldering without too much fear of damage. They are also easily and safely tested with both the vom and logic probe. The actual logic states that exist on the TTLs are shown in the many Test Point Charts throughout the book. The Test Point Chart index (after the Table of Contents) gives the figure numbers of the charts in the book.

THE SENSITIVE CHIPS

The other main group of chips, which include all the large important chips, are too delicate to apply forces to. The forces are pushing and pulling, heat, freezing and voltage. The group is loosely referred to as MOS chips. *MOS* stands for Metal Oxide Silicon; alternately some users might mean Metal Oxide Semiconductor. There are three types of MOS. They are: the NMOS, based around n-material silicon; the PMOS, that has p-material silicon; the CMOS (Complementary MOS), which contains both n and p silicon channels for electrons and holes to move through.

All of the chips in both the rugged TTL and the sensitive MOS groups are encased in ceramic or plastic packages. The packaging has no electrical influence on the chip except that the package supports the metal legs that attach to the printboard or plug into a socket.

Practically all of the chips, which are tiny silicon wafers, are wired into what is known as *Dual Inline Packages*, DIPs. There are two parallel rows of feet—one row on each of the long sides of the rectangular package. More about the DIPs is given later in this chapter. The packaging is the same for both TTL and MOS chips. You can't tell the difference between TTLs and MOS's by looking at them.

While the TTLs are composed of circuits containing bipolar transistors, either npn or pnp, the transistors used in the MOS circuits are the Field Effect Transistors, FETs. The TTL bipolar transistors have elements called emitter, base and collector. The MOS Field Effect Transistors have elements called source, gate and drain. The only thing the TTL chips and the MOS chips have in common is that they are both made out of pieces of p and n semiconductor material.

As described earlier and illustrated in Fig. 4-2, the bipolar transistors in the TTL chip are constructed with building blocks. There are two types of blocks made of n material and p material. A pnp, as the name suggests is a sandwich. The n material

is sandwiched between two pieces of p material. The junctions between the different materials are fused together and form pn junctions. The npn is also a sandwich with p material in the center and n material on both sides. Whether the device is a pnp or npn: the top piece is always the collector, C; the center piece is always the base, B; the bottom piece is always the emitter, E.

The bipolar transistor can roughly be thought of as three gears meshing at the junctions. If you get a small current to flow between the emitter and base, then that will get a large current to flow between the emitter and collector. This is called a current amplifier. The bipolar transistor deals in current amplification. This is different than the FET that deals in voltage amplifying. We'll get to that in a few paragraphs.

The FET is not constructed with the same building block layout. The FETs on the chips are made with a channel. The channel can be thought of as a piece of either n material or p material. Study Fig. 4-6. On the left side of the channel is the connection called the drain, D. The other side of the chan-

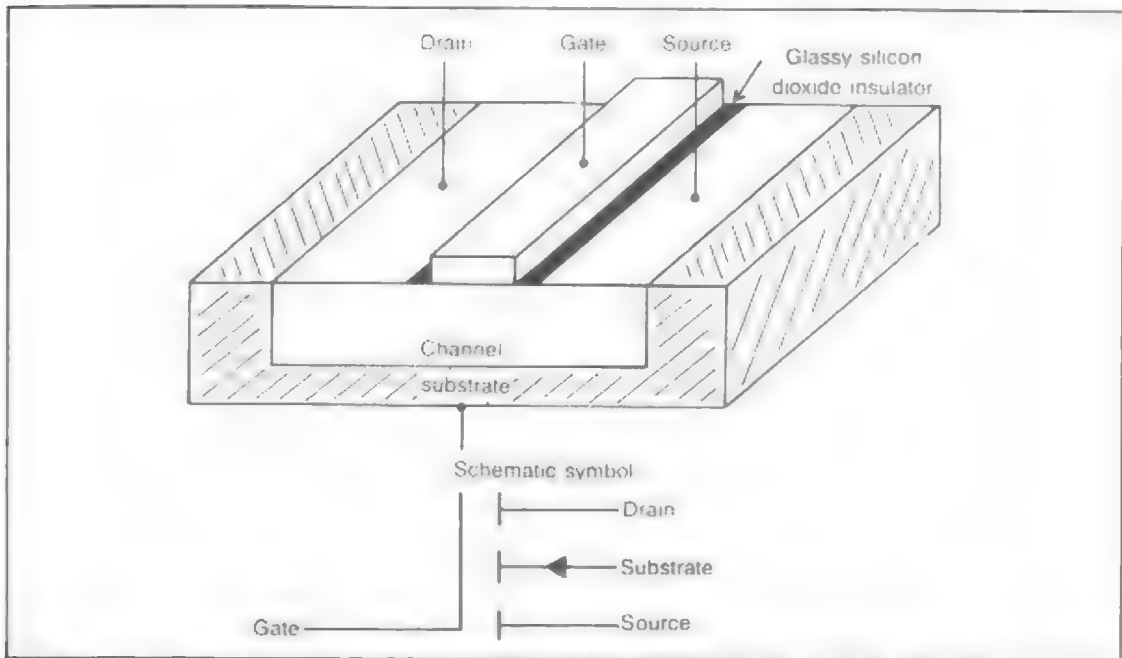


Fig. 4-6. The basic IGFET has four connectable sections, the source, the gate, the drain, and the substrate.

nel has the connection called the source, S. In between D and S is a layer of insulation made of a glassy silicon dioxide. On top of the oxide insulator is attached a metal lead. The oxide and the lead are called the gate, G. Note that the lead is not connected to the channel, it is insulated from it by the oxide.

That the oxide remains intact and maintains the insulation is crucial. When tragedy strikes, as shown in Fig. 4-7, and a hole is blown in the oxide, then the FET will die and your C128 will be in trouble.

Because there is an insulator between the gate lead and the channel, electric current cannot pass from the gate to the channel. In the bipolar transistor, the base, which is analogous to the FET's gate, is not insulated from the emitter-collector path. That is why the bipolar transistor is called a current amplifier. The current from the base controls the emitter-collector current.

In the FET, current can't flow from the gate to the channel, but a voltage on the gate can affect electrons flowing in the channel. A voltage on the gate, if it is varied, in turn will vary the number of electrons that flow in the channel. For example, if the voltage is high, it can stop the electron flow altogether. If the voltage is low, it can allow the elec-

trons to flow in full strength. Whatever its value, the voltage on the gate directly influences the channel electron flow between the source and the drain. That is why the FET is called a voltage amplifier.

During troubleshooting and repair, the way the microscopic transistors are performing in the chips is abstract. There is no ordinary way that you can test the individual TTL or MOS transistors.

All three types of MOS chips plus many variations are in common use. Forms of the NMOS are used a lot in *large scale integration* (LSI). The LSI chips are those with about 100 individual gates on a chip. It is sometimes useful to know that NMOS chips use a + dc supply voltage.

The NMOS is referred to as a single-channel, one-polarity chip. This means that the dc voltage applied to it goes to all the FETs on the chip at the same time. The ground return is also connected to every FET on the chip. But the gates, sources and drains have their own configurations according to the job they perform on the chip.

When the channels are made of p material, the resultant PMOS chip works in a similar way, except that positively charged holes move from source to drain, instead of negatively charged electrons as in the n material. In an n channel, a + voltage is ap-

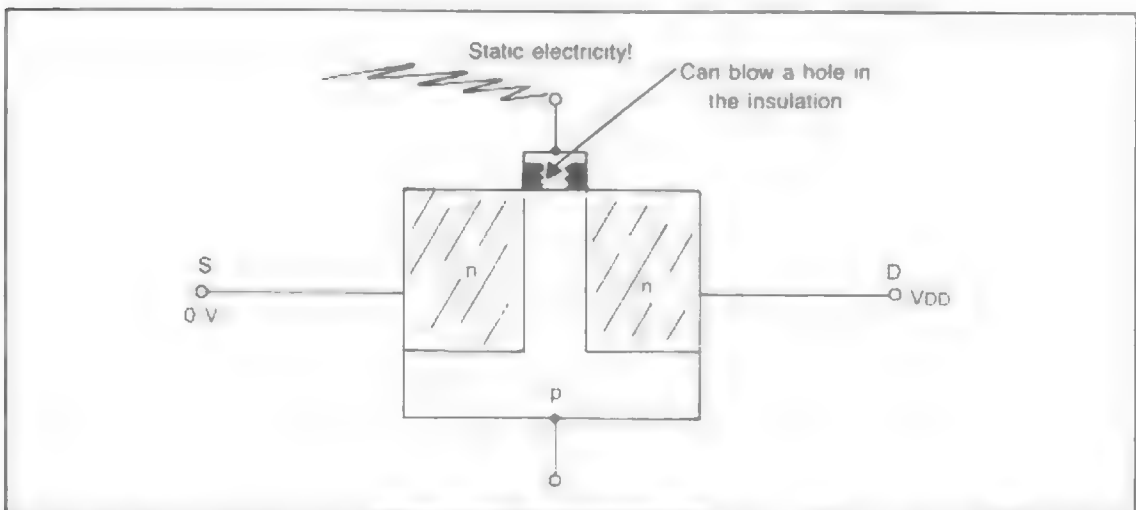


Fig. 4-7. The glassy oxide insulator between the gate and the channel is very fragile. Static electricity can easily blow a hole through it.

plied to the drain to attract electrons from the source. In a p channel, a negative voltage is applied to the drain to attract holes from the source. The gate still does the voltage controlling job except that it varies the intensity of hole conduction rather than electron conduction. You'll recognize a chip with p channels because the schematic shows a negative dc supply connected to the chip. Whatever the polarity, whether holes or electrons are on the move, the sensitivity of the MOS chip-to-gate oxide rupture remains the same. Great care must be taken while handling and testing any type of MOS chip.

The CMOS chip is the common one used in *small scale integration* (SSI). SSI is the term used for the chip with less than 10 gates to a chip. The CMOS type is also used a lot in *medium scale integration* (MSI). MSI chips contain between 10 and 100 gates. Please note that the number of gates is not the same as the number of individual transistors. There can be many microscopic transistors in a single gate.

Typically the supply voltage to a CMOS is of a positive nature. Internal wiring takes care of changing the voltage polarity and applying a correct polarity to the different channels. The NMOS is thus able to propel electrons from source to drain. The PMOS is able to move holes from source to drain. The insulated gates are then able to exercise control over the channel currents, no matter whether electrons or holes are on the move.

Just as the TTL chips have been designated numbers in the 7400 or 74LS00 series, the CMOS small package chips are assigned numbers in the 4000 series. An example of CMOS chips in the C128 are U2 and U20; both are type 4066, a quad bilateral switch. This chip receives more attention in Chapter 8.

THE DIP PACKAGE

If you take a close look at the chip layout on the board of your C128, then you will see that the chips are in a rectangular package with two rows of evenly spaced feet on the two long sides of the rectangle. There are no feet protruding out of the top or the bottom of the chip. In the Test Point Charts, the

chips are drawn exactly as they exist. On the schematic the chips are drawn for the convenience of the drawing. On the schematics the connections bear no physical resemblance to the actual chip as the Test Point Charts do. While you can take test readings directly by comparing the Test Point Chart to its chip on the board, you can't do that with the schematic. You must take an extra mental step when using the schematic. You have to relate the pin numbers of the schematic to the pin numbers on the physical chip. This extra step clutters the troubleshooting reasoning you are going through. It takes time to master relating the schematic to the physical circuit. With a little practice though, the technique can be handled and become natural.

The C128 printboard is full of chips. Practically all of them are DIPs, Dual In-line Packages, as mentioned earlier. On each chip, there are two in-line rows of tiny feet. The top view of the chip reveals a rectangle with a key designation at one end. The key is usually a notch, sometimes a paint dot or an indentation. When a chip is replaced, the keyway of the new chip must be placed in the same way the chip of the old one was. These DIPs will fit into either its socket or its holes in the printboard, backwards. Don't install the chip backwards!

The pins all have numbers on the Test Point Charts and on the schematics. To find pin 1, look to the left of the keyway, with the keyway at the top of the rectangle. In the C128, a lot of the chips are mounted upside down. The keyways are then at the bottom of the chip as it lays on the printboard. In those cases, start your count to the right bottom and count up.

Whichever way the chip is positioned, count counterclockwise around the chip. The last pin on the chip will be across the keyway from the number 1 pin.

When you replace a DIP, whatever you do, make sure the key is in the same position as the original was before you put in the new one and energize the computer. The key is only a visual indicator. The chip will fit into the socket or printboard holes wrong or right. Make sure it is right. You can double-check the key position on the chip location

guide. On the guide, note all the chips that are installed with keys at the top and the others with keys at the bottom.

As you test the feet, which are test points on the chip, you must be able to read the numbers on the chip. The numbers are not marked. You must be familiar with the numbering system as just described.

Tests with the C128 energized consist mostly of applying the vom probe to detect the voltage, touching down on the test point with a logic probe to find out what logic state the pin is in, and touching a test point with a low impedance probe on an oscilloscope. With the plug pulled and the C128 off, then the main test is with a low voltage continuity tester. The resistance between a pin and ground is the most popular one. Other resistance tests to check continuity between test points are also used.

The fastest way to find a pin on a chip is by knowing at a glance how many pins are on a chip. In the C128 there is a wide assortment of chips with different numbers of pins. The largest chips have 48 pins. The rest have less. Use the 40-pin chip in Fig. 4-8 as an example. Pin 1 is at the upper left of the rectangular top view. Pin 40 is opposite to pin 1 across the keyway. At the bottom left is pin 20. Across from it is pin 21. At the center of the chip on the left are pins 10 and 11. Across from them at right center are pins 31 and 30.

With a little effort, you can quickly train yourself to find all six pins rapidly. Once you have your eye on those pins, then you can touch down on any other pin using those original six as reference. If you are going to make a lot of tests on different pins on the chip, then you can take a couple of toothpicks and stick one between 10 and 11 and another be-

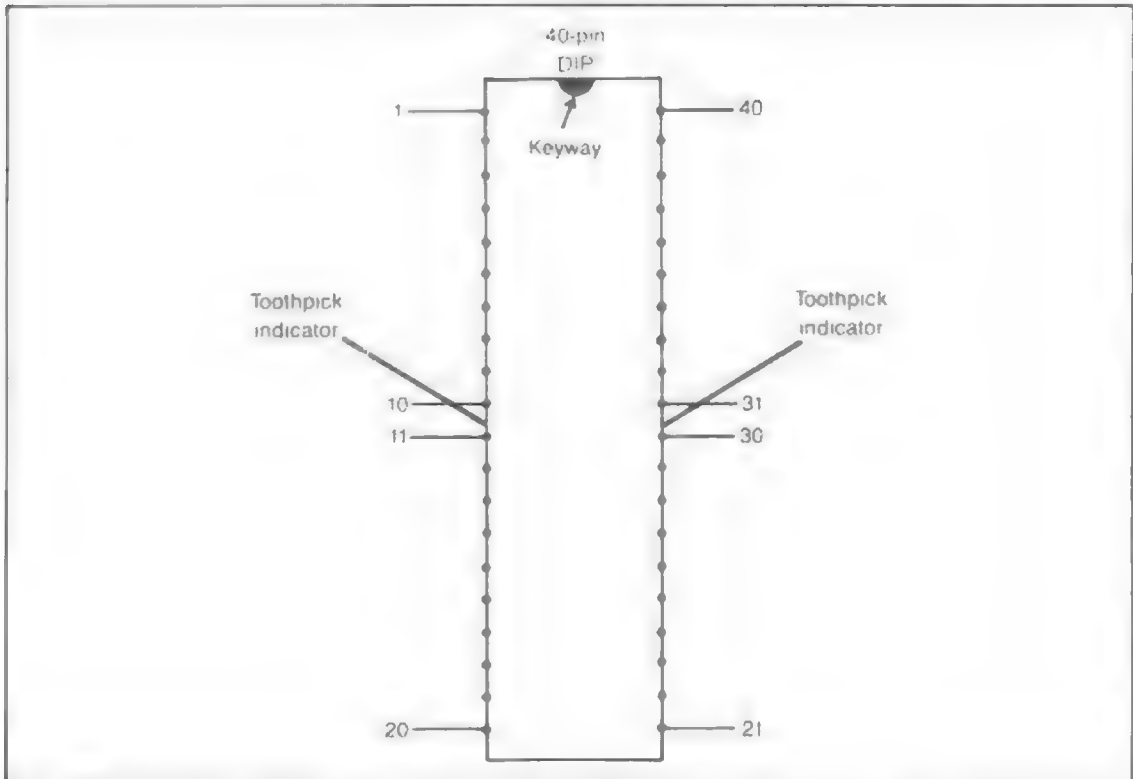


Fig. 4-8. Counting pins on a large chip can be made easier with two toothpick indicators inserted at locations 10-11 and 30-31

tween 31 and 30. That way you'll save a lot of long counting. The most you'll need to count on this 40-pin chip is four pins. For instance, if you want to test pin 25, you can count from pin 21—four up. Pin 36 can be located by starting at pin 40 and counting four down. That way, if you are making a lot of careful readings, you won't accidentally touch down on the wrong pin.

Other chips you might have to handle have different numbers of pins emerging from the package. You could find small DIPs with 14 or 16 pins. Other packages have 18, 24, 28 or 48 pins. The 48-pin types are the largest in the C128. No matter what the number of pins, the general packaging arrangement, and counterclockwise numbering around the keyway, are all the same.

Printed on the DIPs, easily seen with the naked eye or with the help of a magnifying glass, are all sorts of servicing and replacement information. There are many parts houses in most neighborhoods where the C128 is found, such as Radio Shack. The markings on the chips in your C128 are helpful when it is time for you to purchase a new chip.

Note the markings on the chip in Fig. 4-9. All the marks have meanings. First of all, note the logo of the manufacturer. It is a familiar Commodore sign. Following the logo is the chip part number. The 6526

is the generic part number. Often the manufacturer will put his own part number on the chip instead of the generic number. Table 3-4 is a list of all 63 chips in the C128 and their replacement part numbers.

With the number off of the chip you want to replace, any parts house clerk can cross-reference that number to the part number that they carry.

Next on the chip is the code date. This is supposed to be a deep, dark secret but it is easy to read the date on most chips. This one says 4186. This means that the chip was manufactured in the 41st week of 1986. Others might be more tricky, but use your imagination. What you puzzle out will probably be correct. This code date is especially useful during the time the C128 and its chips are under some sort of warranty.

There is one word of warning when purchasing replacement chips. If you are changing a chip try to get an exact replacement, that is, original manufacturer's parts. Of course, if that is not possible then you must try another brand name. When you do install a replacement from a different manufacturer and the trouble still remains, or a new trouble starts happening, double-check the new replacement before embarking anew on the troubleshooting trail. Sometimes you can get a supposed exact replacement that is not exactly exact.

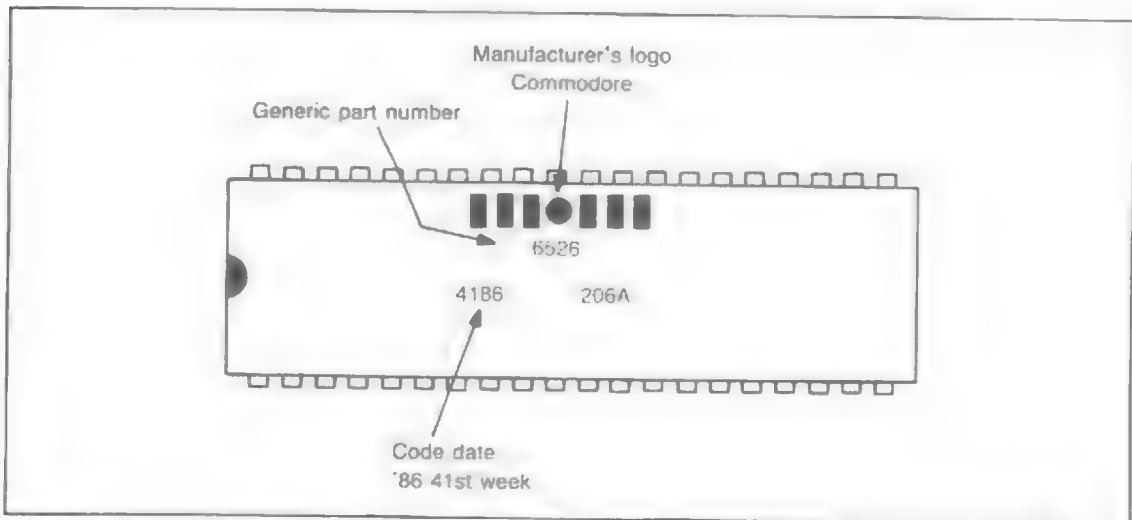


Fig. 4-9 The markings on chips have meanings. You must use these markings when ordering replacement chips.

As you look down the list of 63 C128 chips in Table 3-4, there are many TTLs in the 7400 and 74LS00 families. There are a couple of CMOS chips in the 4000 family. Then there are a lot of chips not so easily found on generic lists. The C128 has the two MPUs, the 8502 and the Z80, the 8721 PLA, the 8722 MMU, the two video output chips, the video controller 8563 and VIC 8564, the two CIAs both 6526's, the 6581 SID, the ROMs and the RAMs.

When you suspect these chips and are seeking a replacement, your first stop should be at a Commodore service agency. Next you can try some of the national mail order houses. They often come up with odd chips.

The most expensive replacement chips you will encounter are the ROMs. When you purchase them you are buying software that is burnt into hardware. The ROMs contain the C128 operating system program and other vital programming. There is more about the ROMs in Chapter 7.

SOCKETED CHIPS

The C128 has some of its chips in sockets. For example, the two 6526's and the 8563 under the metal video box shield. The four main ROMs are in sockets along with another empty socket that can hold a fifth ROM. SID is in a socket as is the 8722 MMU; this totals twelve sockets. All the rest of the chips are wired into the printboard. I must add that this is the arrangement in my C128. Different production runs could alter the layout.

The removal and replacement of the socketed chips is relatively easy. All you need is a few guidelines to ensure that the job will proceed in a safe and smooth fashion.

Chip Removal

You probably won't be able to help yourself, but a good rule to follow is: never touch a chip with your hands, body or clothing. Act as if you have a deadly communicable disease that you can kill the chip with. TTLs are not as sensitive as the MOS chips, but I'd treat them in the same way. That is because, on occasion, you might pick up what you think is a TTL and it turns out to be a sensitive RAM MOS.

The reason for the antiseptic approach is that you can be carrying a static electric charge that could kill a chip by electrocution.

The weak, vulnerable time of an MOS is when it is loose. Any static spark into the chip at this time could very easily burst some of the insulated gates of the tiny FETs. On the other hand, when the chip is plugged into a circuit, it is no longer vulnerable. It can take a lot. When chips are shipped from a manufacturer, they are plugged into a piece of conductive foam. This effectively ties all the pins together, and the chip is no longer vulnerable. As you handle a chip keep it in its conductive foam pad till the instant before you insert it into a socket or the printboard holes.

The chip you remove from a circuit should be handled in a safe manner too. Often the chip is good and will have to be put back after testing and handling. Small chips, those with 24 pins or less, can be extracted and handled safely with the DIP extraction tool shown in Fig. 4-10. The tool is simply a specially built type of tweezers. It is made with two little lips that can be placed under the two ends of the chip. This allows you to gently rock the chip out of its socket. Once out, the chip can be placed on a conductive grounded surface that shorts all the pins to the surface. With all the pins shorted to ground, no static voltage can build up and kill an FET gate.

The DIP extractor tool comes with a small hole on its top. This is to allow you to screw a grounding strap onto the tool. The grounding strap is then attached to earth ground through a one megohm resistor. One important word of caution. Do not ever remove or insert chips while the computer is plugged into ac, whether it is on or off.

The extraction tweezers work well with chips of 24 pins or less. It can be used with the larger pins, too, if you take some extra care. The longer DIP bodies, especially the 40 and 48-pin types, will be placed under physical stress if you use the same technique as you did with the smaller chips. The way around that problem is not to pull the chip all at once. Take one side at a time. Gingerly rock the chip out of the socket, first tugging one side and then the other. That way the holding ability of the socket is gradually relaxed and the large chip will not ex-



Fig. 4-10. The chip extractor is the safest way to remove chips.

perience undue strain. Once again, after the chip is free, place its feet on a conductive surface.

Chip Replacement

During troubleshooting and repair, chips are usually in place when you begin. The preceding section went through chip removal from a socket. There comes a time in repair when the chips must be inserted back into the socket. Either a new replacement or the old, proven good, chip is to be reinserted. Just as much care must be exercised during the chip insertion as the extraction.

The chip to be inserted should be standing on a conductive grounded surface. You could use the grounded extractor tool if you are surehanded and careful. There are dangers though as the little feet are fragile and getting all the legs into the socket at the same time, without bending a leg or two under, is tricky. Therefore, it is advisable to use the chip inserter shown in Fig. 4-11.

The insertion tool in the illustration is typical of the devices that do this job. It has a conductive post sticking out of the top where a grounding strap can be attached. The post connects to all the metal parts

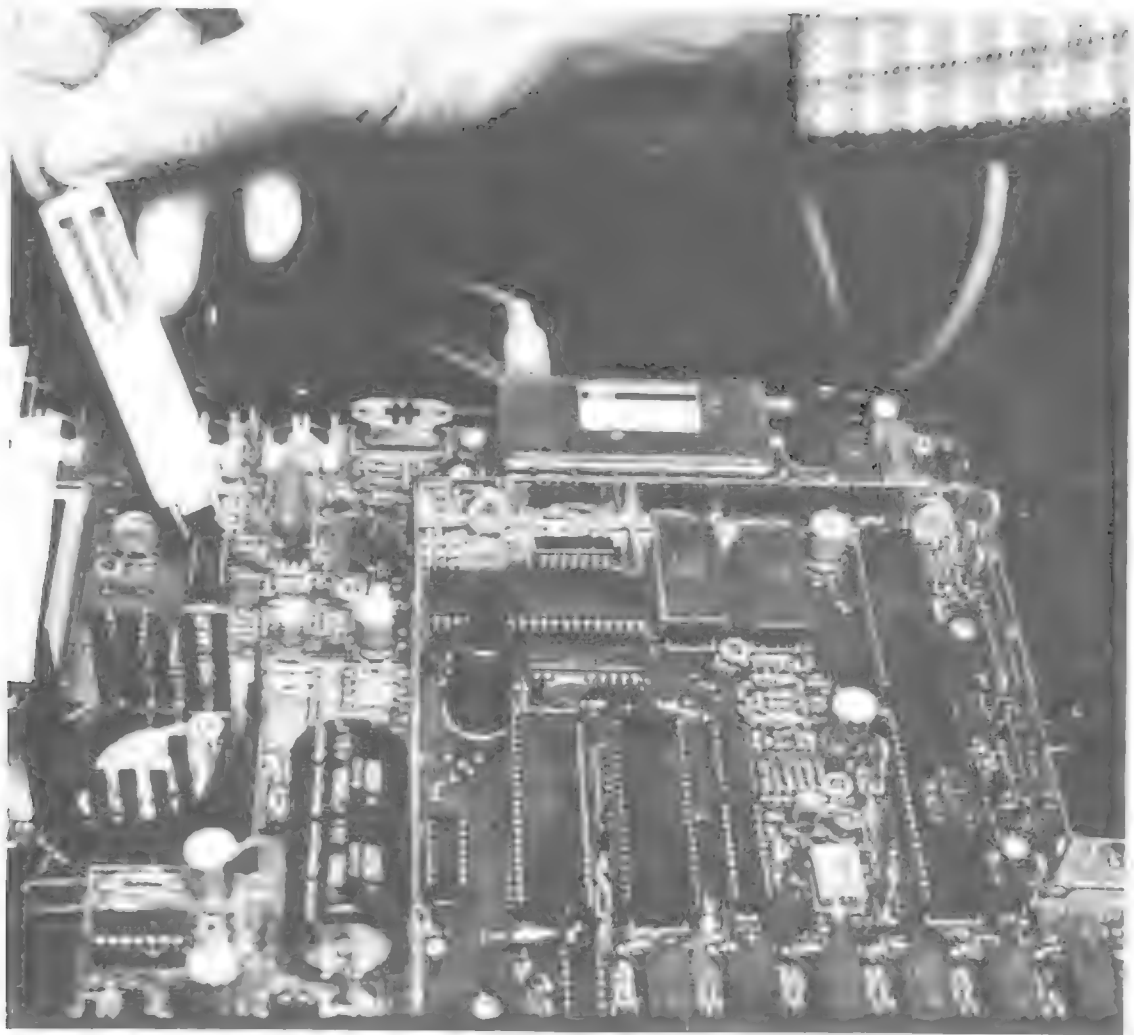


Fig 4-11 Chips are best inserted with a gadget that can hold and keep the chip grounded at the same time

of the tool. There are two metal holders at the bottom of the tool that open and close. The holders are able to grasp a chip or let it go as you pull the post up and down. On the side of the gadget is a locking button. This can lock the holders and post so a chip can be held firmly. That way a chip will not be dropped accidentally during an insertion.

A feature of this inserter is a pin straightener, also seen in Fig. 4-11. The pin straightener is also grounded to the post. Often the little legs of a chip can be squeezed out of line. By simply pushing the

chip into the grounded straightener, all the legs are lined up properly. I usually make sure of the leg lineup on every chip I handle whether it looks like it needs it or not. Let's go through the chip insertion step by step.

To begin with, the chip should be standing on its own feet on a conductive surface. In this position the entire pinout is at the voltage level of the surface, which is zero volts or ground. Look closely at the pins. Are they all lined up nicely? Are any of them bent?

If any are, or just on general principles, pick up the chip with the grounded extractor tool. Push the chip, feet first into the grounded pin straightener on the side of the insertion tool. Rock the chip gently till the pins are all in the best in-line spacing. Then place the chip back on the conductive surface. Do not use the extractor to insert the chip. The inserter is better suited for the job, as you'll see.

Pick up the insertion tool. The post should be jumpered to earth ground. Pull the post out. The twin holders on the bottom will respond by retracting. Place the holders over the subject chip and release the post slowly. The holder will now come down snugly around the chip and ground all the pins. Note that the extractor had held the chip by its insulated ends. The inserter holds the legs instead and grounds them at the same time. Meanwhile, the chip had also been grounded on the conductive surface it was sitting on. You can now lift the chip off of the conductive surface. The chip remained grounded during the procedure. First on the conductive surface and then to the holders of the inserter, the chip was never off ground and was never vulnerable to static sparks.

Once the inserter is in control of the chip, then the chip can be placed over its socket. The legs are carefully placed onto the top of the socket holes. Observe the keyway on the chip and on the socket. They must match so that the legs will enter the correct socket holes.

Once the pins are all lined up and making contact with the socket, pull the post up carefully. The holders will release the chip. Again the chip was not out in the open and vulnerable. The pins were transferred from the ground connections on the inserter to the circuitry connected to the socket. Then with the post still held up, press the inserter against the chip and seat it firmly. Do not press too hard—just enough for proper seating. Then remove the insertion tool.

The extractor doesn't keep the pins at ground level. It grasps the chip at its insulated ends. It holds on to the plastic packaging material of the DIPs. The inserter, on the other hand, grounds the pins. Grounding is the trick to keeping the MOS chip intact during handling. If you must move or manipu-

late the chips out of circuit, make sure you are personally grounded. Attach grounding straps to your wrist. Should you have to transport chips from place to place—even just across the room—keep the chips in a grounded condition (e.g. plugged onto the foam pad it comes boxed with). It is very trying to order a chip, have it arrive after a week or so, and then lose it to a shot of static electricity, as you carry it across the carpet on a low humidity day, when all you had to do was keep the chip grounded.

SOLDERED-IN CHIPS

Of the 63 chips in the C128, only 11 of them are in sockets. All the rest are soldered directly to the printed circuit board. This presents a problem when one or more of them must be removed or reinstalled. The desoldering and resoldering of chips has been described as a job for an artisan, not the ordinary person.

Perhaps this is true, if you want to reproduce the same finished look that is accomplished by robot machines in a factory. However, as much as you take pride in your work, the polished look of a finished job does not influence the operation of the computer at all. The only important thing is replacing the chip accurately so that the computer begins operating once again.

Desoldering

If you do find yourself faced with a chip soldering job, it is wise to consider taking the C128 to a good computer repair company and letting them do the honors. However, if you do feel confident that you can handle the chore, read on. I'll explain the technique.

It is not that difficult, but you must take your time and examine every move after you make it. The potential for inducing additional problems is a large factor. It is bad enough to have one trouble in a C128. Great care must be taken to avoid causing a second and third complication by careless techniques.

The first step is to reach for the right soldering iron. Only the right one will do. Don't place a hot 100/400 watt bench gun against the printboard. It

is much too hot. Use the lowest wattage iron you have. One that will just about melt the solder. Thirty watts is the absolute maximum, and if you have an iron with less wattage use it. The iron should be one specifically designed for chips and sensitive transistors. They are on the market—some are battery operated and others use dc operating schemes. These are good to use when replacing MOS chips because the use of dc eliminates all the 60 Hz line voltage sine waves you get out of the house sockets. However, if you do use a conventional ac house current with an ordinary low wattage iron, then just ground the iron like you did your wrist. Of course, it goes without saying that the C128 should not be plugged into any house sockets during soldering operations. The C128 must be completely unplugged: power supply and keyboard—this means printer, disks and everything.

The trick to using solder on chip legs is control of the heat. Too much heat, even momentarily, or prolonged heat from a low wattage iron can kill a chip. Too little heat does not let the solder form a good joint. A cold solder joint between the chip leg and the board can produce intermittent and otherwise difficult additional troubles. All surfaces that receive solder must be clean, and the tip of the iron should remain tinned all during the operation. *Tinning* an iron tip simply means keeping a thin coat of fresh solder on the tip. The tip can be wiped off with a piece of cloth periodically. Wipe quickly so the cloth is not on the tip long enough to start smoldering.

Heatsink techniques are a must. The best heatsink is to grasp the lead between the connection on the board and the body of the chip with very skinny long nose pliers. That way as you apply heat to the connection, the heat runs up the leg to the pliers and is tapped off before it can get to the body of the chip.

If you are not adept enough to hold the iron, the solder, and the pliers all at the same time, then an alternative heatsink can be rigged with a piece of lamp cord. Attach a small clip lead to the cord. Clip it onto the spot where the pliers would have been holding. The rest of the cord can hang loose. The heat will always take the path of easiest dissipation,

which is the skinny plier nose or the lamp cord rather, than the tiny leg on the chip. It is a good idea to ground the pliers or the lamp cord too. This prevents the connections from developing any undesirable static buildup.

As mentioned, a solder tip is easily kept clean and tinned by wiping it with a cloth or paper towel. Wipe quickly to avoid charring the towel. The solder must be rosin core, and it is advisable to use type 60/40 (tin to lead). It melts at 371 degrees Fahrenheit.

Desoldering a chip is not too hard, especially if you are sure it is bad and you do not care if it is damaged. Even if you do care, the technique when properly performed, is easy. The first step is to position the board in a convenient place, where you are not off balance while working. The top and the bottom of the board should be free and clear of obstacles. A good bright bench lamp with a magnifier would be extremely helpful. Get into a comfortable position so that you are not straining. Then the job can begin.

Note the number of connections. The job will consist of freeing each connection in turn and removing all excess solder. Attach the lamp cord heatsink to the first connection—take them each in turn. Don't rush. Realize that it is going to take time. Touch the hot iron to the connection. Most of the heat that is melting the solder flows into the lamp cord. Jiggle the chip leg with a solder pick and wipe the connection. Keep doing that patiently till the pin is free. It will take a number of picks and wipes till you get the feel of it. When most of the solder is removed and the leg is free, go on to the next connection. Pin after pin is freed in this way. Keep working till each individual pin is free. Then lift the chip off the board. Should some solder drip across the board while you are working, stop. Clean up the drip before proceeding.

There are many soldering devices available that will make the job a bit easier. One device is a solder sucker. It is a soldering iron complete with a rubber suction ball. Place the solder sucker over the hole, with the ball squeezed, and as the solder melts let the ball inflate. The solder will be sucked into the ball system. With a little practice, the solder sucker will speed up your desoldering time.

An experienced technician, when he is sure the chip is defective and he just wants to get the chip out of there, uses less care. He simply applies heat and pries. After awhile the chip pops out. Remember though, he is experienced and has taught himself shortcuts. I wouldn't advise moving so quickly till you have successfully replaced a lot of chips.

Resoldering

Once the old chip is out, then clean up the holes in the printboard. Use a bit of heat, some wiping, and pick out all the excess solder. All you want left are tinned open holes. The new replacement then can fit nicely into the holes. With the new chip in place, apply a drop of properly heated solder to each connection. After checking the connections for possible shorts or opens, then the job is done. Just be sure you don't put the chip in backwards; it will fit, and then you'll have the entire desoldering-resoldering to do all over again.

As you can see, the tough part of the job is the patient loosening of each lead and the preparation

of each empty hole during the desoldering process. It is good technique to never have to desolder the same chip twice. Often a particular chip will die repeatedly due to a manufacturer's design error. The chip, over the life of the computer, fails over and over again. You will do a lot of muttering if you find yourself desoldering and resoldering the same chip every so often. What can you do? There is an easy solution. You can avoid the unpleasant chore after the first replacement. Make it a rule in your repertoire of soldering techniques to always install a socket whenever you have to remove an unsocketed chip. That way you'll never have to resolder that particular chip again.

Follow the same resoldering instructions for installing a socket as you would for a chip. A bonus for installing a socket instead of the chip directly to the printboard is, you need not worry about hurting the socket as you solder it in place, like you would with a chip. You still want to use as little heat as possible though, since the printboard is still sensitive, as well as are the adjoining chips.

5. The LSI Chips

If you look down at the exposed printboard of the IC128, (further revealed with the chip location guide of Chapter 3), then you'll see: two large 48-pin DIPs; four almost as large 40-pin DIPs; one DIP with 28 pins. Should you remove the metal shield with the rows of ventilation holes, you'll find two more of the 48-pin variety. These nine chips are classed as Large Scale Integration. LSI chips contain at least a thousand individual microscopic circuits. These germ size circuits that are placed on the chips are, under ordinary circumstances, inaccessible to us humans. Perhaps a microbe could crawl in and out of the circuits, but we can't get to them to run tests. Furthermore, even if we could somehow run a voltage or scope test and find a bad circuit among the thousand or more circuits, there would be no feasible way to remove the defective component and install a new replacement.

Therefore, these thousands of circuits on a chip, from a troubleshooting point of view can be thought of as one. When a chip is deemed bad, it is replaced, not repaired. Fortunately there are a number of

techniques that can be used to determine if a chip is good or bad. First of all, the pins that stick out are all test points. A probe can make definitive readings that can be compared with what is supposed to be on the pins. Secondly, you can contact and test the registers of the circuits in the chip by writing to them with a POKE or reading them with a PEEK.

If you consider the number of circuits in comparison with the number of pins, then there could be 1000 circuits accessed by 40 pins. This means, on an average, one pin connects to at least 25 separate circuits. Each one of the tiny circuits could have 10 input-output leads. That makes each pin on the DIP a connection to 250 internal nodes. While this statement has many variations and complications, the point is: all the circuits inside the chips exist in a different microscopic world. Therefore, during troubleshooting and repair, forget the internal circuits of the LSI and larger chips, and consider each chip as a black box with 40 or so test points available to your vom, logic probe, scope and continuity tester. Only the original designer and manufacturer

know what is inside the black box down to the last detail.

Fortunately, to service the C128, it is not necessary to learn all there is to know about its LSI chips. You do not need to know the detailed construction secrets of the chips. The servicing skills required are in general those techniques that are used to repair any electronic circuit. The servicing techniques are those that specifically apply to the chips in the C128. In this chapter, there is an overview of the nine chips: two 6526's, the 6581, 8502, Z80, 8722, 8721, 8564 and 8563. This is an acquaintance you'll need in order to approach the chips. Later in the book is a full chapter devoted to each of the chips, replete with servicing techniques.

THE MICROPROCESSORS

The C128 is three complete computers in one, and they share some of the circuits. The C128 and C64 modes share the 8502 MPU. Each mode uses the 8502 as its MPU—at different times. Both modes cannot run at the same time, but you can switch from one mode to the other quite easily. The 8502 is an upgrade of the 6502 found in the VIC 20 and the 6510 in the C64. All three MPUs use the same instruction set. Commodore purposely chose to go this design route so that as much software as possible will be compatible in the three machines.

The Z80 MPU is used expressly for CP/M duty. It is brought into play when you place a CP/M disk in the drive and start up the C128. When the Z80 goes into action, the 8502 is placed on hold. The CP/M software library has tens of thousands of programs that you can use with the Z80 doing the heavy work. The Z80 and the 8502, even though both occupy the C128, have very little in common.

The two processors share the same bus lines at different times. It's as if two strangers use the same bedroom, but one works graveyard and the other works daylight. They never contact each other, but do share the same facilities. The 8502 is really the main occupant of the C128 and the bus lines are designed to handle its input-output directly. The Z80 can't directly use the lines designed for the 8502. Therefore, the Z80 has to use a special interface circuit when it takes over the bus lines while

the 8502 is not operating. All this will be discussed in more detail in Chapters 12 and 13.

An MPU is commonly referred to as the heart of the computer. It is called the "heart" because it connects and works with all the rest of the important chips. Like a heart it has no brains but plenty of strength. The brains of a computer are programs in the ROMs. These programs are known as the operating system. The MPU can be likened to a main telephone exchange. It is connected to all the registers in the chips of the memory map. The memory map is like a telephone directory. In the C128 128K memory map, there are 131,072 addresses. The MPU is connected to all of the addresses.

The MPU has four types of lines that connect to the chips with the addresses. They are the address lines, data lines, control lines and special I/O lines as seen in Fig. 5-1. The MPU in all computers is vital. It performs the computing. Without an MPU, a computer is not a computer. You can design a computer and leave out any of the other chips. You cannot leave out the Processor and still have a computer. At this time let me introduce the general duties of the MPU. They might not have much meaning at this stage, but the duties will be clearer as we go.

The MPU, first of all, has the job of providing and requesting data. Figure 5-2 illustrates both of these jobs. The reason it requests data from the rest of the computer is to process it. The data is obtained from places like the RAM, ROM and the CIAs. The MPU will send a message to one of these places and request data. The data is sent back to the MPU over the data lines. The MPU can then process the data and send the finished data back to one of those storage or I/O chips just mentioned. How does the MPU alert a data holding area that it wants data?

The message that the processor sends to a memory map location is an address, like a telephone number. The address goes out over 16 address lines in either the 8502 or the Z80, whichever one is in charge of the computer. The address lines are shown in Fig. 5-3. The 16 lines carry the address. Each line can send a high voltage or a low voltage. A high is a binary 1 and a low is a binary 0. The address consists of voltages that mean 1's and 0's. The

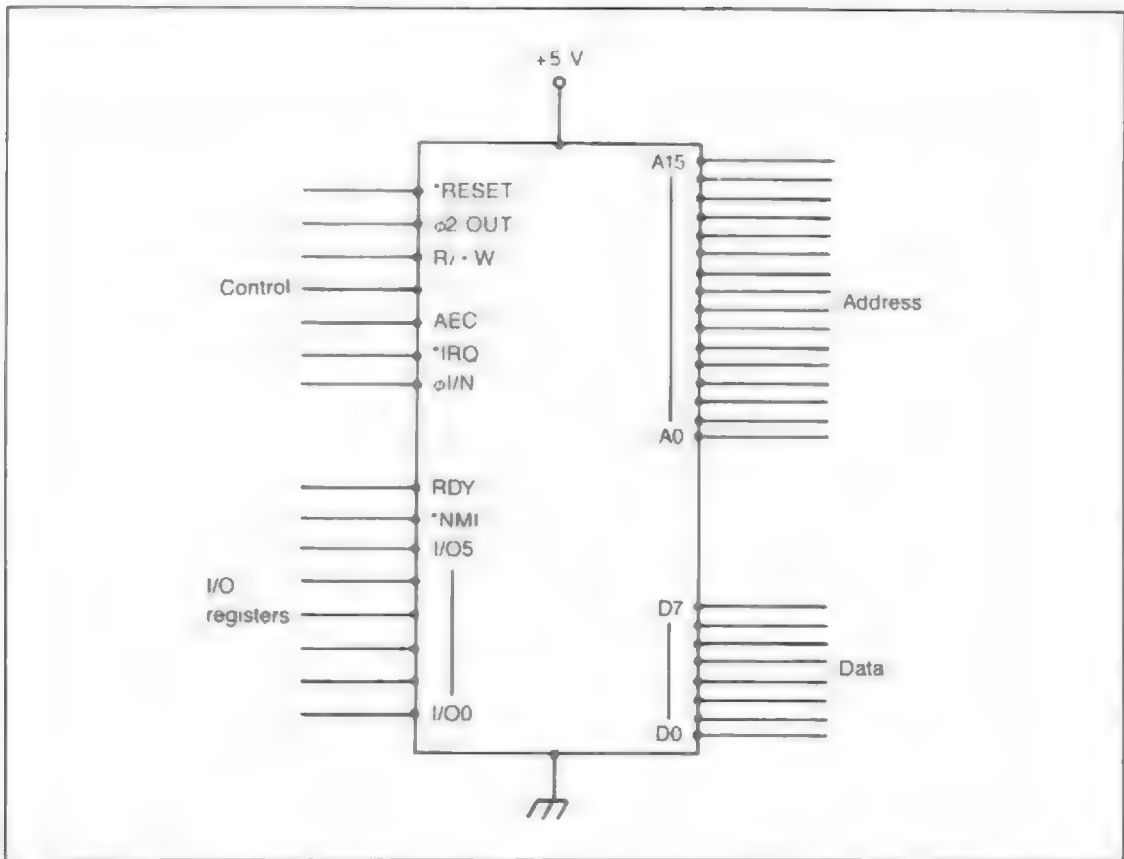


Fig. 5-1 The 8502 processor has four types of lines that connect to the rest of the computer chips. They are the address, data, I/O and control lines.

address consists of 16 bits. One address bit is transmitted out over each address line. The address lines are one way avenues. The bits go out but none come back, just like when you dial a friend on the telephone. Their number only goes out, it does not come back.

Sixteen bits can form 65,536 different patterns of 1's and 0's. Each different pattern dials up a different location. Because both the 8502 and the Z80 only have 16 address lines, and if they do not get any additional help, then they can only address 65,536 (64K) of the total 131,072 (128K) locations in the C128. But they do get help. The 8721 PLA chip and the 8722 MMU chip give the MPU address lines a hand. With their help, the 128K memory map is broken into two 64K sections and the MPUs are able

to address each 64K section in turn. This is discussed further in this chapter and throughout the rest of the book.

When a processor in the C128 needs data, it outputs, on its address line, a set of 16 bits that opens up a location just like your telephone line is opened up when someone dials your number. As soon as the C128 location has opened the data in that location, then a copy of the data in that location moves out on the data bus, as shown in Fig. 5-4. The data then travels to the processor that called for it. If the 8502 called for the data, then the data traverses the data bus and enters the 8502 without further ado. If the Z80 had called for data however, then the data travels the bus to the Z80 area. There, the data is met by an interface consisting of a latch

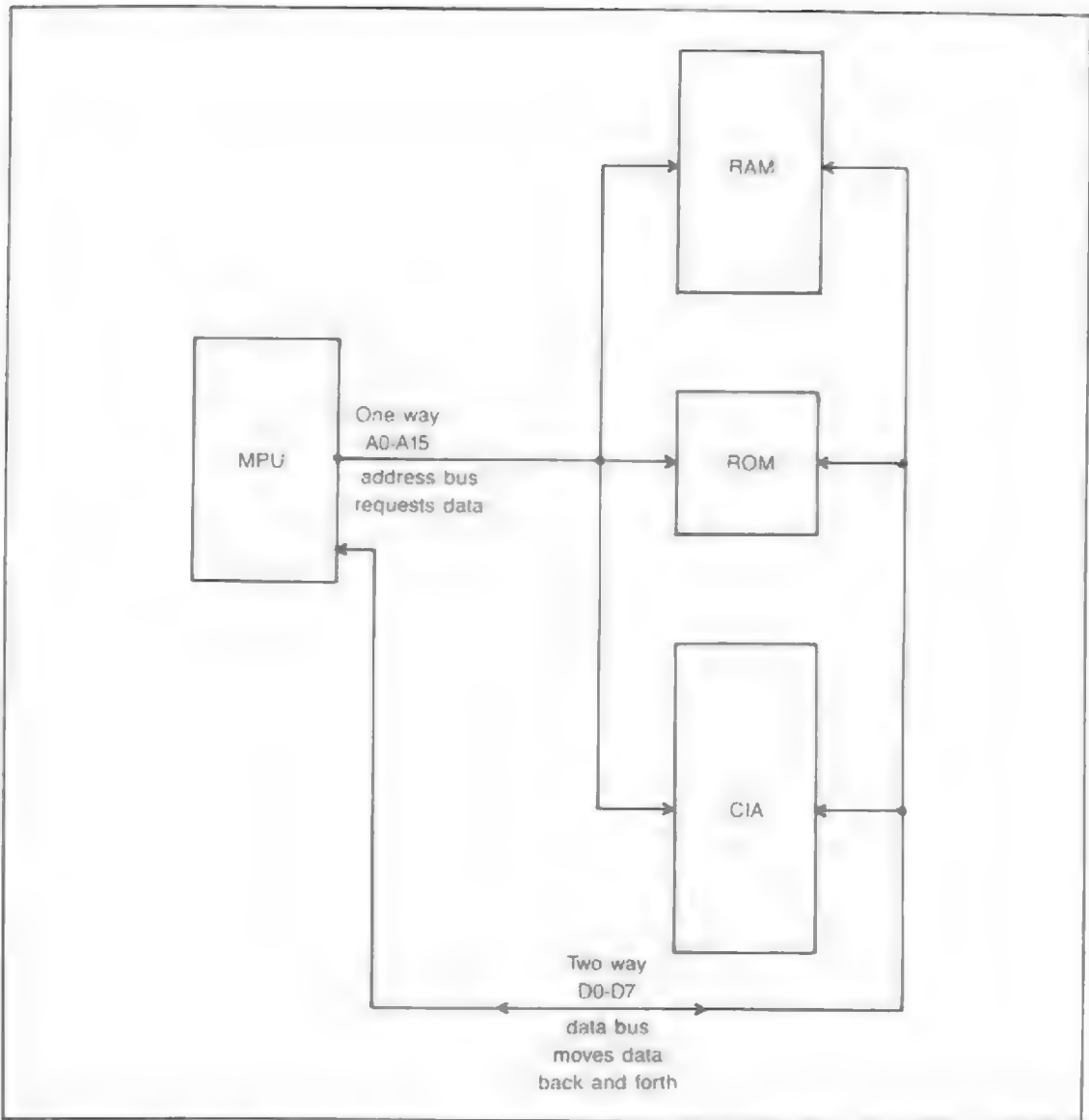


Fig. 5-2. The address lines have the job of going to a location and opening it up so that the data in the location can be accessed. The data then can travel between the MPU and the location

chip and a buffer chip. The data moves through the two chips and from there can enter the Z80.

Once inside the processor, the data is manipulated. An MPU is capable of doing some limited mathematical and logical processing of the data. The nature of this work is covered in Chapters 10, 11,

12 and 13. Once the data is processed, then the MPU is ready to send the data back to storage or to an output peripheral.

The MPU then addresses the location where the data is to be sent. The addressed location is activated and opened by its own combination of address

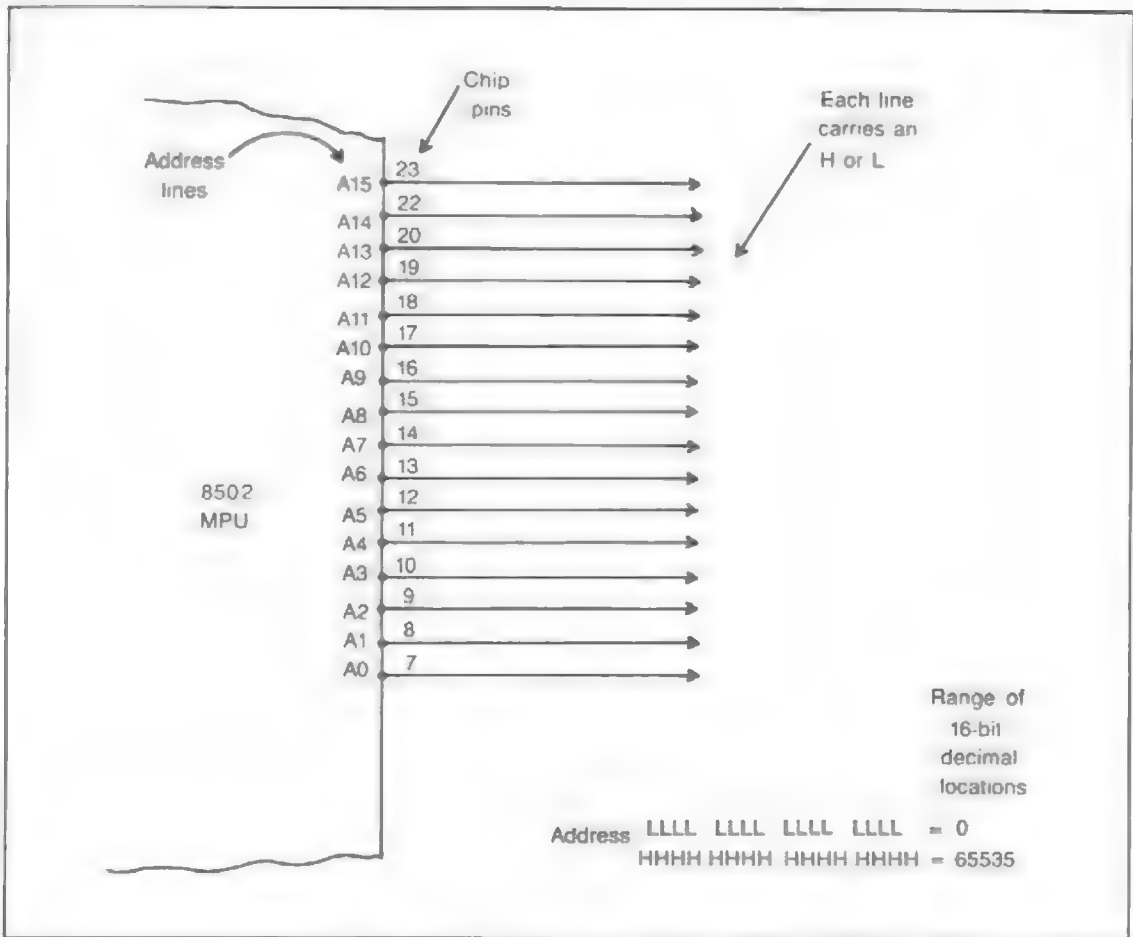


Fig. 5-3 The 16 address lines are able to carry 65,536 different combinations of highs and lows.

bits. The MPU then sends the processed data out over the same data bus line. Note that the address bus is one way from MPU to location. The data bus on the other hand is two way. It moves data from location to MPU and then after processing back to a location. If the location is RAM, then the data is stored. If the location is ROM then the data in ROM is read. When the location is an I/O port then the data is readied and output to a peripheral.

The two C128 processors are both eight-bit. Eight-bit refers to the number of lines in the data bus. Remember that the address bus is 16 bits wide. Eight-bit also refers to the usual size of the RAM, ROM and I/O registers. The chip registers are eight-

bit (one byte) wide. In the eight-bit processor, the data is usually moved in byte-sized pieces.

The rest of the lines in the processors, excluding the address or data lines, are called the "control bus." They are, however, not really bus lines because they are all individually operated and all do different types of jobs. They are, for example, the reset line, clock lines, the R/W line, interrupt lines and others. They will be covered in more detail in Chapters 12, 13 and others.

This explanation briefly describes what the MPUs in the C128 do as they work through a program. In old time "computerese," the job has been known as the fetch and execute cycle. When the cy-

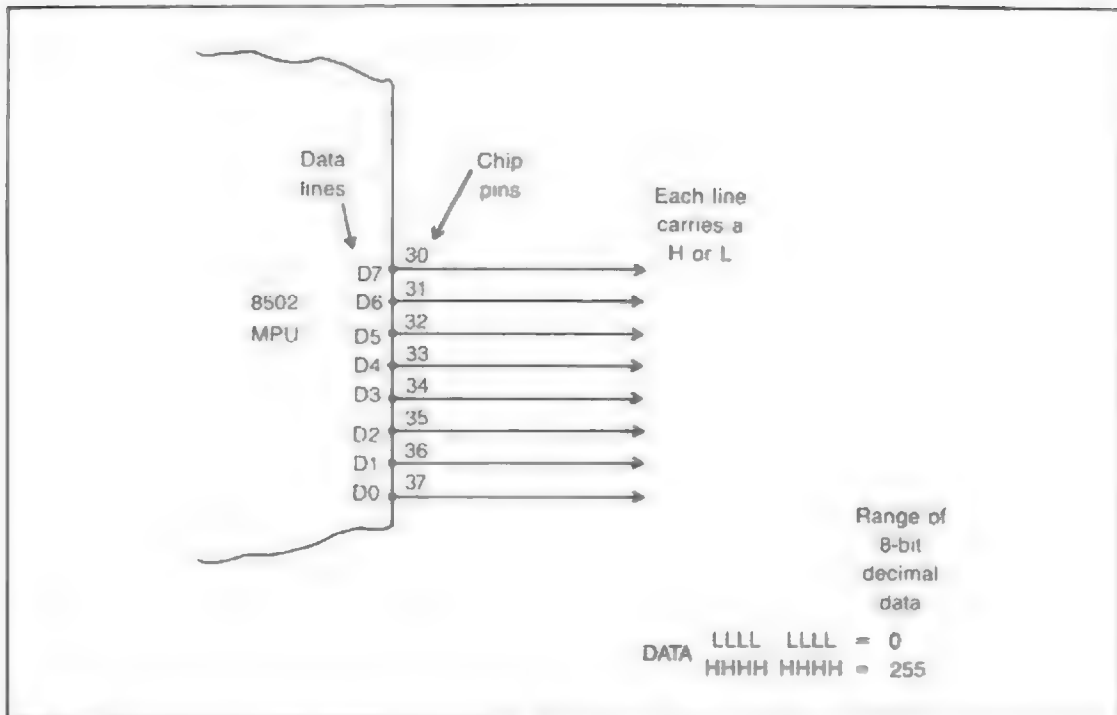


Fig. 5-4 The eight data lines are able to transport 256 different binary bit combinations.

cle breaks down, then the computer needs troubleshooting and repair services. It is a must for you to understand this hardware layout and general operation to apply the fix to the majority of repairs. Your ability to write programs will not aid very much here.

THE 6526 COMPLEX INTERFACE ADAPTERS

The two CIAs are mounted on opposite sides of the board. U1 on the right side of the board is mounted near the port for the keyboard to plug into. U1 is placed there since its main job is to apply the ASCII bits that are generated when you strike keys, from the keyboard into the digital circuits—specifically the data bus. The CIA on the left side of the board, U4, is mounted there to handle the ports over on that side. For instance, U4 is the I/O port for the nearby user port, as Fig. 5-5 demonstrates.

The CIAs can act as either an input or an output chip. They interface inputs such as the keyboard

and joysticks, input-output machines like the cassette and disk drive, and output-only units like a printer.

The CIAs connect to the MPU through the same data lines that go to all the other locations in the computer. The CIAs have their own locations, like RAM and ROM although not nearly as many. When either the 8502 or the Z80 desires to access a CIA, it outputs the CIA address and is connected directly.

The right side CIA is almost exclusively devoted to the keyboard and the two control ports just above the keyboard. A CIA is equipped with 16 output pins, divided into two sections of eight pins each. The keyboard uses all 16 pins to input your key strikes. The control ports each need four pins to input their voltages, as seen in Fig. 5-6. There is no conflict, however, because both of them are open circuits when they are not in use. Since the control ports are not usually used while the keyboard is, they can share lines and they do. While the keyboard is not

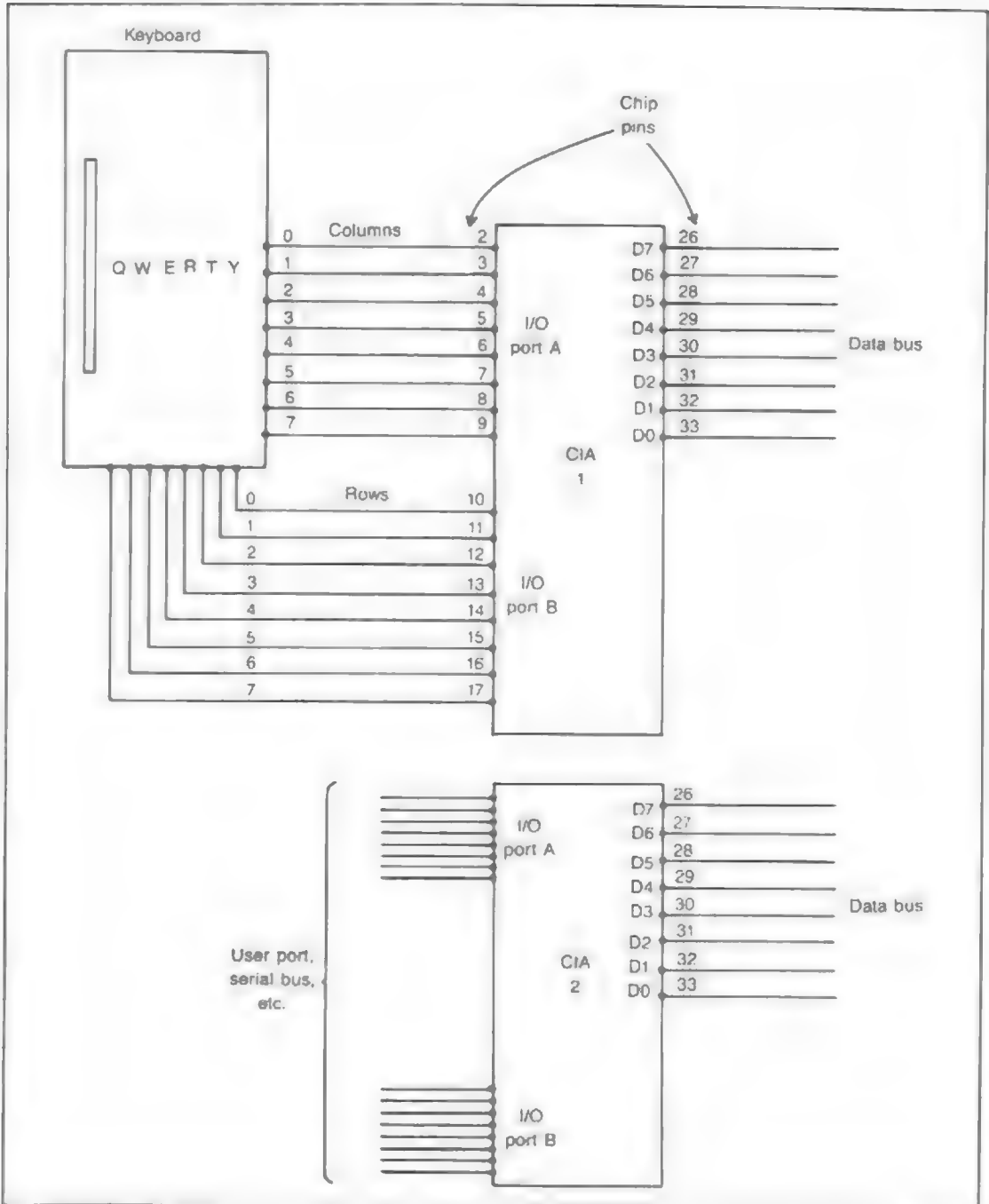


Fig 5-5 CIA1's main job is to interface the keyboard rows and columns to the MPU. CIA2 performs the I/O duties required to interface external peripheral devices to the MPU.

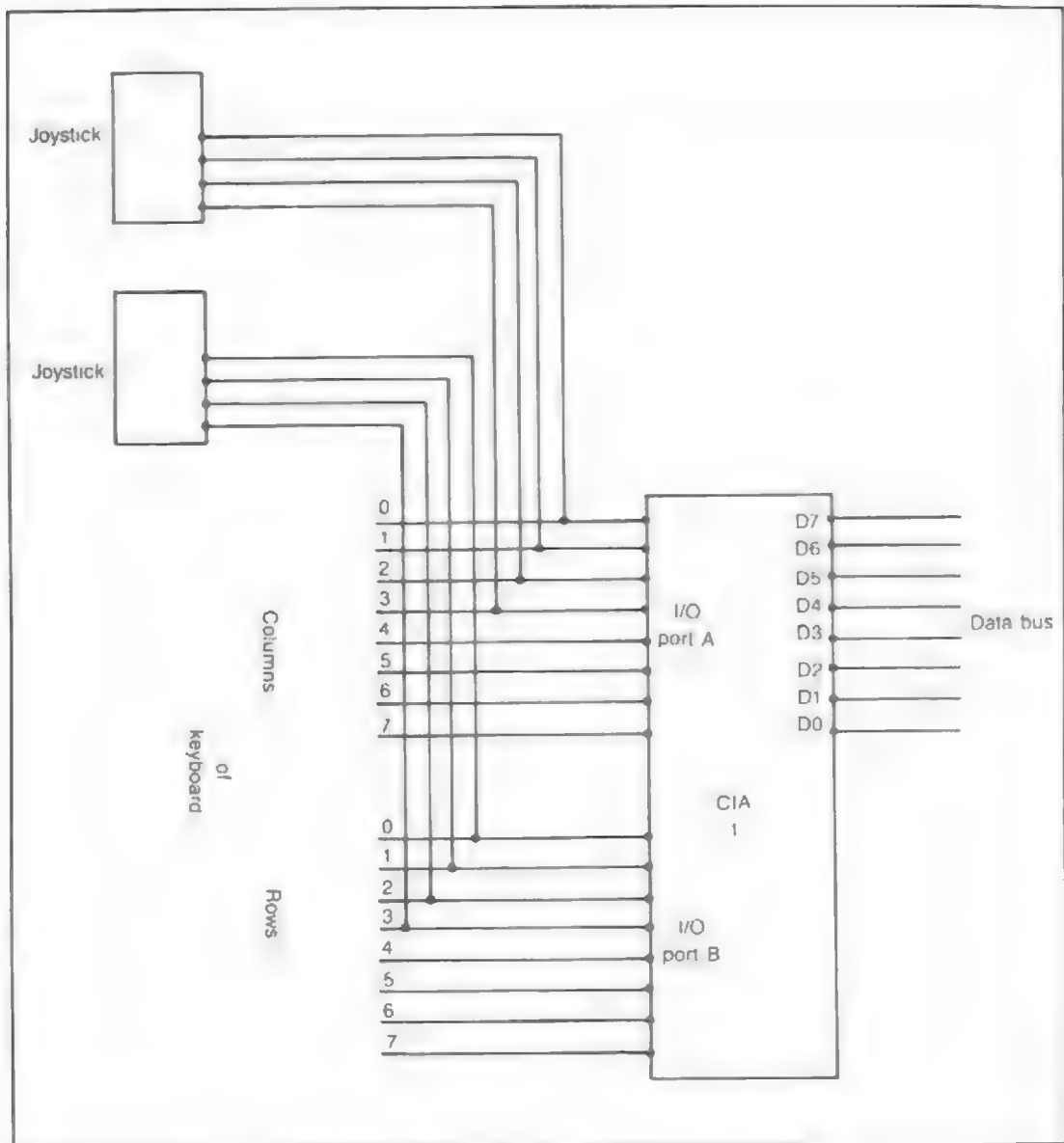


Fig. 5-6 CIA1 also handles the joystick inputs over the same lines that the keyboard uses.

being used you can use the joysticks. Just avoid using them both at the same time.

When you hit a key on the keyboard you are shorting out a row and column. Internally the keys are wired in a block composed of vertical rows and

columns as seen in Fig. 5-7. There are eight rows and 11 columns. This produces 88 row-column intersections where a switch can be placed. In addition to the 88, there are also four more singly wired switches for the RESTORE, 40/80, SHIFT LOCK

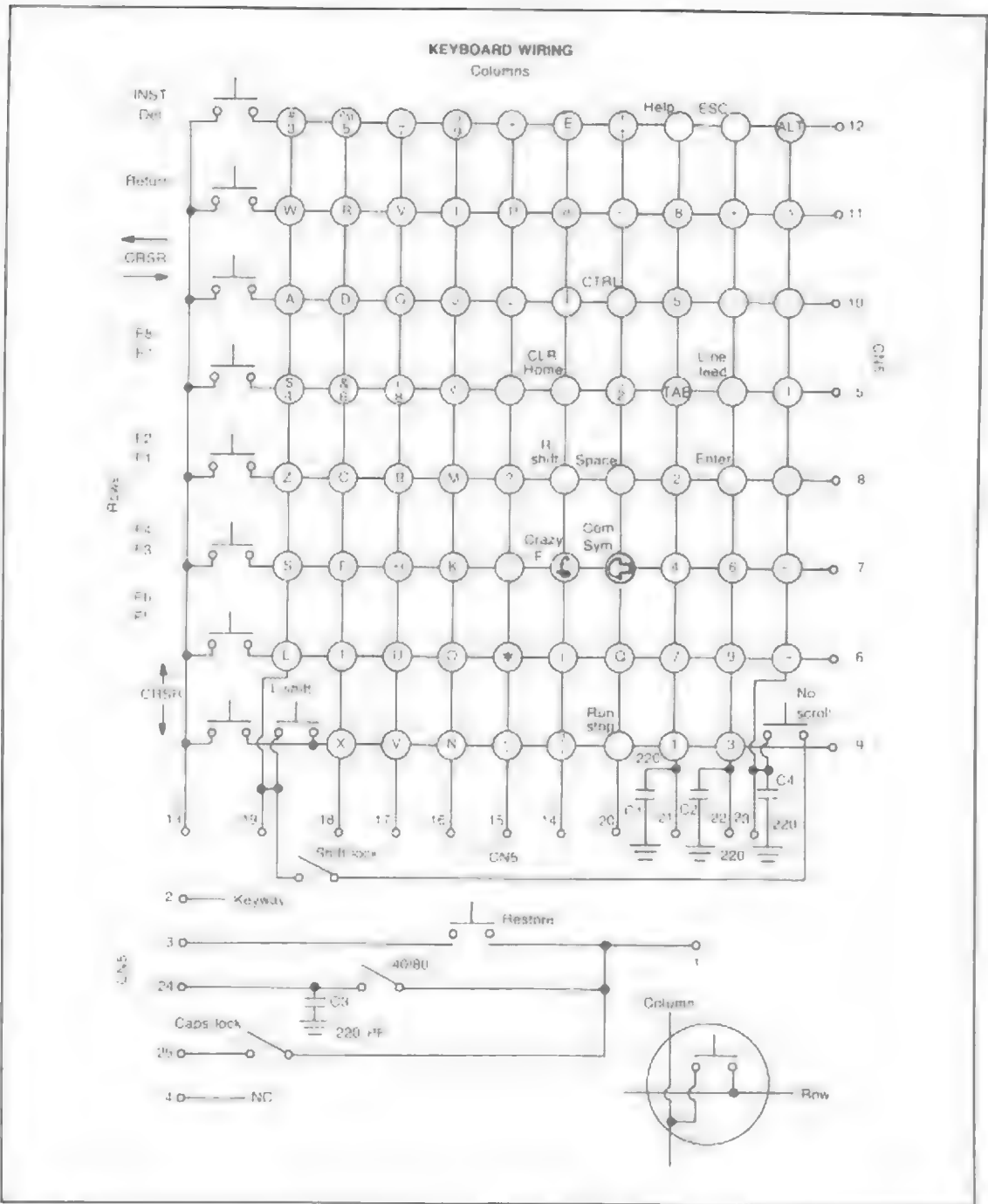


Fig 5-7 This is the schematic diagram of the keyboard in your C12B. When a key is pressed either a switch like the RETURN key is closed or a row is shorted to a column as seen in the bottom right inset.

and CAPS LOCK switches. There are 92 keys on the keyboard including the numeric pad that duplicates the numbers, the positive sign, the negative sign, and the decimal point. The ENTER key on the pad duplicates the RETURN key on the main board.

When an individual switch is shorted then the circuitry generates a character inside the computer. The keyboard CIA as shown in Fig. 5-5, is the device that transfers the shorted switch identity to the MPU so that the desired character is generated.

The CIA on the left side of the board, performs the same kind of duty for other external devices like the User Port and the Serial Bus. These ports are able to connect to a printer, a disk drive, a telephone modem or even another computer. With additional software, the CIA will be able to send or receive from a large number of devices.

The 40 pins of the CIA are shown in Fig. 5-8. There are 24 pins to handle data. There are eight pins that connect to the internal computer side and

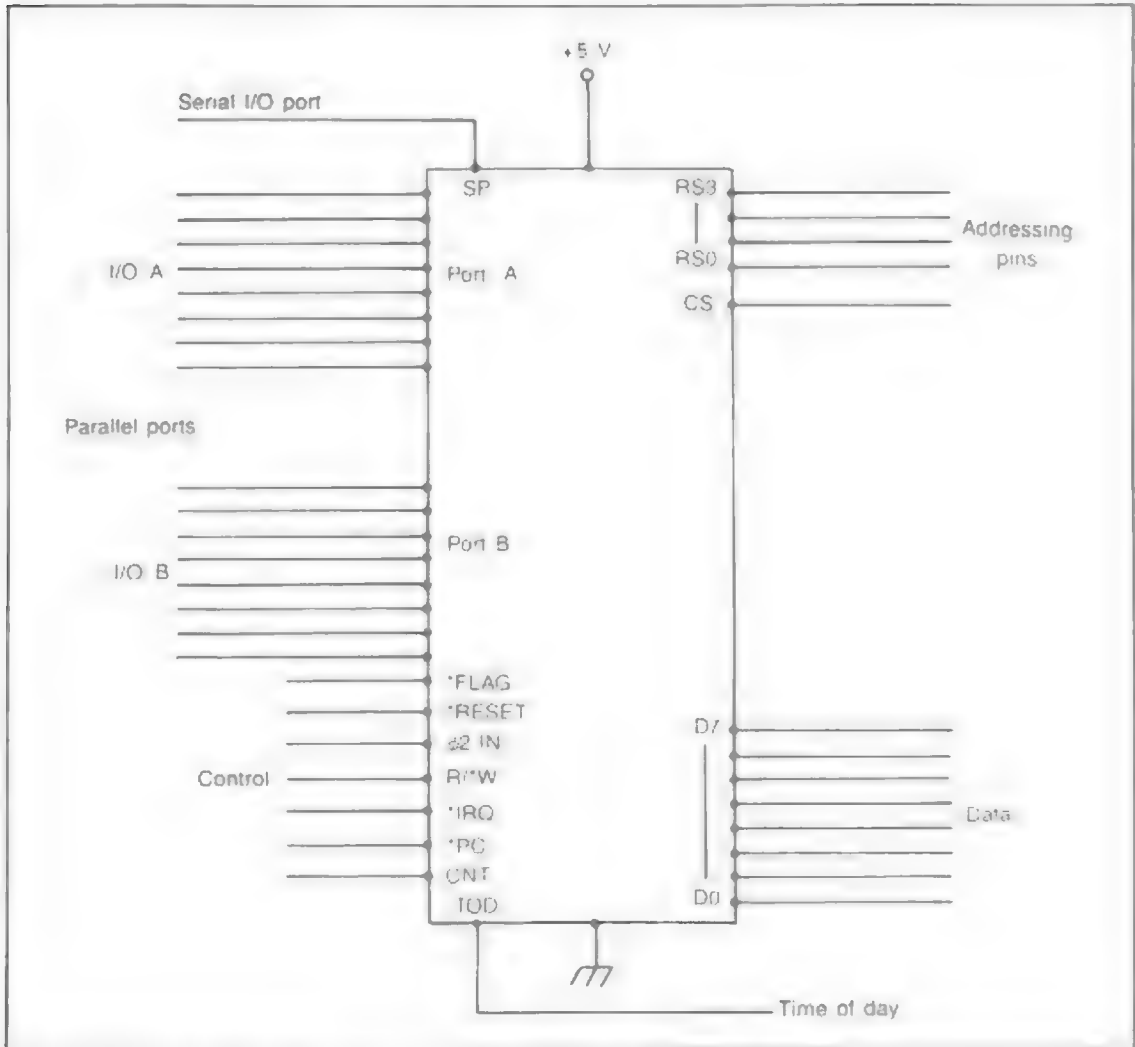


Fig 5-8 The CIAs have address, data bus connections, control lines and two eight-bit ports

16 pins that hook up to the external peripheral side. The 16 pins as mentioned are broken into two separate port sections of eight pins each. These ports are discussed in detail in Chapter 19.

The rest of the CIA pins are used to address the chip and control it. There are five lines into the CIA for addressing purposes. The CIAs only contain 16 locations—unlike the RAM and ROM chips that could have thousands of locations on a chip. The five address lines handle contacting the 16 CIA locations easily.

In addition to the normal I/O duties that the CIAs perform, they also have a 24-hour *Time Of Day* (TOD) clock which can be set for both AM and PM. There is also an alarm for each clock. You can set the time and alarm with a few program lines.

For some sophisticated computer operations, special timing mechanisms are required. In the CIA, there are two such circuits called 16-bit interval timers. They are said to be independent and linkable. These are characteristics needed when these timers are used. There is more detail about these timers and the TOD in Chapter 19.

Another feature of the CIAs is a special eight-bit serial I/O shift register. The term I/O means *input-output*. The CIA has its 16 output pins broken into two 8-pin ports. Because an eight-bit port outputs eight-bits at a time, the eight-bits are all moving at the same time abreast, and that is called *parallel output*. This bit movement is in contrast to a series of bits that are moving out of one pin, in single file. The single file movement of bits out over one line is called *serial output*. You have probably heard the terms parallel and serial concerning the transfer of digital bits of information. Figure 5-9 contrasts this type of data movement in the CIA.

The CIAs each have (in addition to two eight-bit parallel ports) one serial I/O port using one pin. In the chip, connected to the pin, are circuits that make up an eight-bit serial register. The eight-bits that the register can hold will leave or enter through one end of the register, get shifted from bit holder to bit holder in single file, and exit or enter the chip at the other end of the register. Chapter 19 has more detail on this operation.

The CIAs can also act as the middleman during

peripheral-computer handshake operations. The handshake is so named because, as a data transfer operation, many verification checks of the data occur while transferring. The handshake is a complicated procedure whereby the computer can either receive information from a peripheral (read) or send information to a peripheral (write). The handshake is vital if the data is to be transferred error-free.

The handshake is a check and double-check between the computer and the peripheral. If we endow computer and peripheral with human characteristics, then the handshake will proceed as shown in Fig. 5-10. For instance, a peripheral can be attached to a CIA with information it wants to transmit. The peripheral says over the lines, in digital code, "Hi there 8502, you old MPU. I have some data for you."

The 8502 receives the message via the CIA. The 8502 replies through the CIA, "I hear you calling. All is clear, send the data."

The peripheral promptly sends the data over the parallel lines and waits. The data enters the CIA, passes through the CIA and exits into the internal data bus. The data then travels the data bus and enters the 8502. The MPU notes the data entry into its registers and calls to the peripheral, "Okay there peripheral, the data is received, I'm ready for more."

The peripheral thus sends information to the MPU and is able to continue sending until all its data has been transferred correctly. Although the characterization of the two devices is exaggerated, the procedure is essentially correct. The CIA chapter has a description of the electronics that handle the handshake operations.

THE VIC AND VIDEO CONTROLLER CHIPS

Under the metal shield are the two 48-pin video output chips. They are both running all the time, but do not necessarily output the video. They do output a border and a display block. The one that is chosen to run will output the video.

The 8564 VIC is simply an upgrade of the previous VIC chips found in the C64 and the VIC-20, as shown in Fig. 5-11. Like its predecessors, one out-

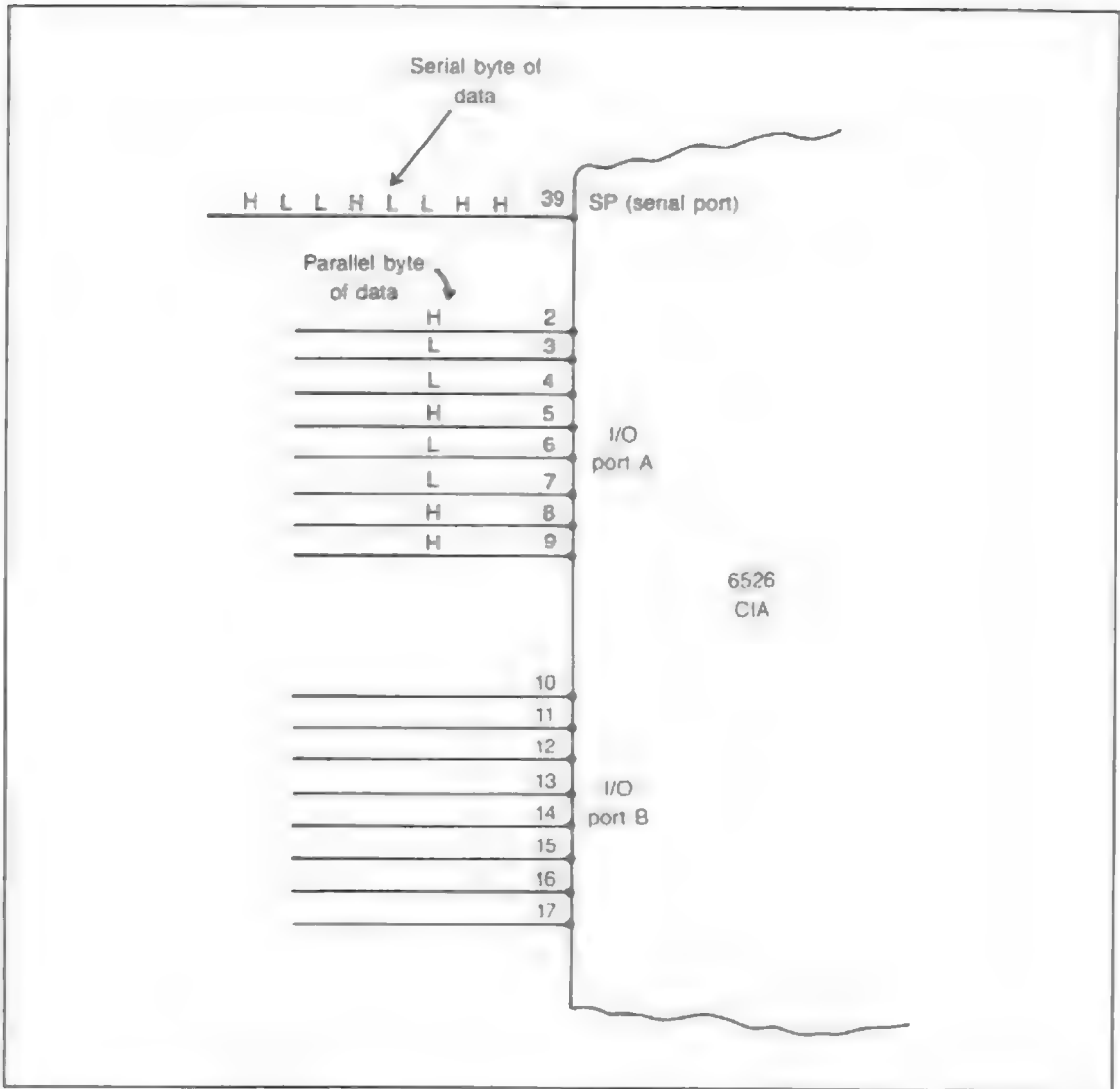


Fig. 5-9. The CIAs are equipped with two forms of ports. First are the two parallel eight line ports, and second is a single line serial port.

put is a display block containing 40 columns across and 25 rows down to total 1000 character spaces, as shown in Fig. 5-12. The 8564 sends the video in two pieces from two pins called SYNC/LUM and CHROMA. These signals are applied to the RF Modulator box. When you are in the 40-column mode—whether it is the C128, C64 or CP/M—you are using the output from the 8564 VIC.

The other 48-pin chip in the metal enclosure is the 8563 Video Controller. It is a new chip and provides other sorts of video output.

The 8563 sends video in four pieces. These parts of the total video exit out of four pins, as shown in Fig. 5-13. The signals are called R, G, B and I. The R, G and B obviously mean red, green and blue. The I stands for *intensity*. Intensity has to do with

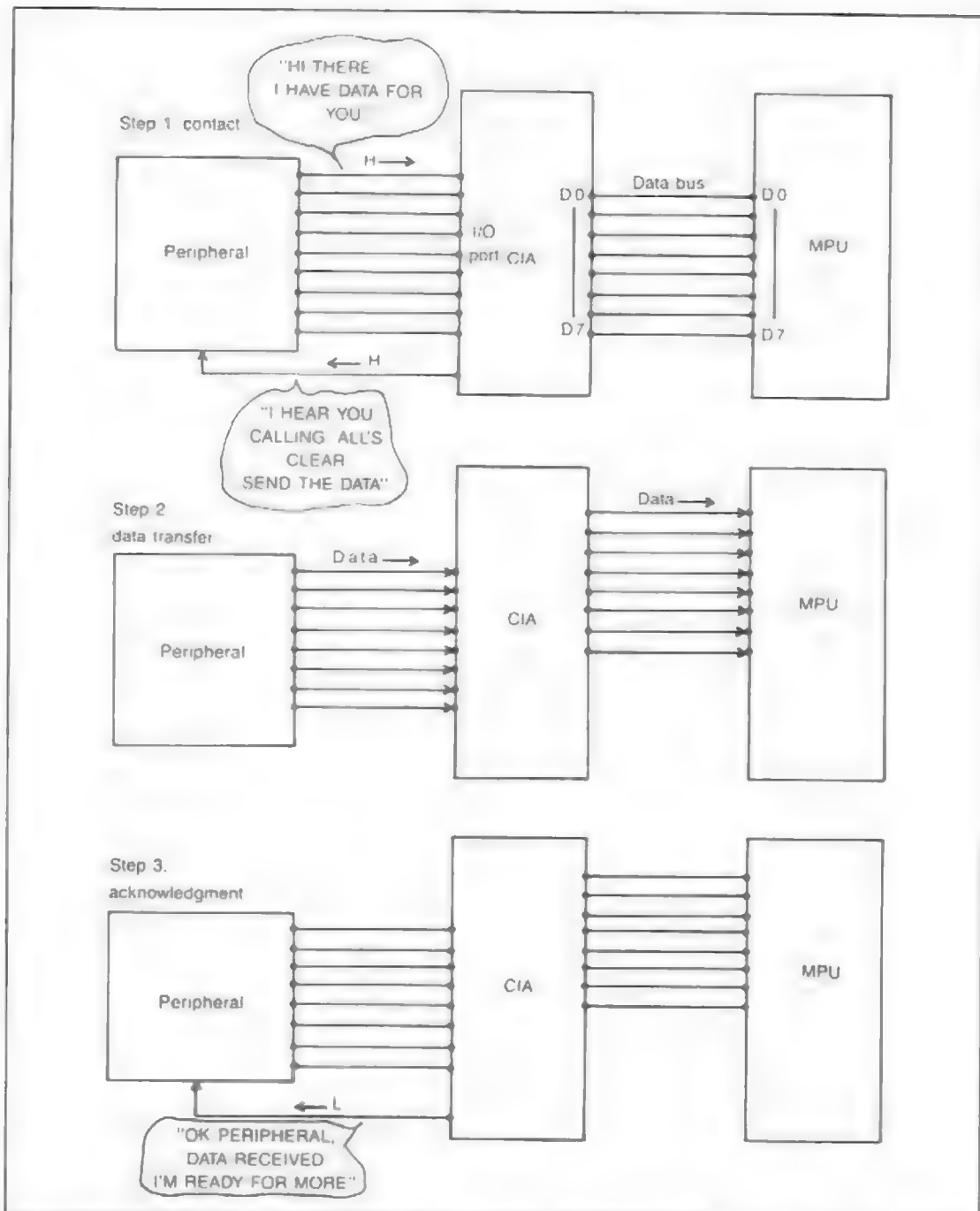


Fig. 5-10. The CIAs are the interface units that permit a "handshake" operation between a peripheral and the MPU.

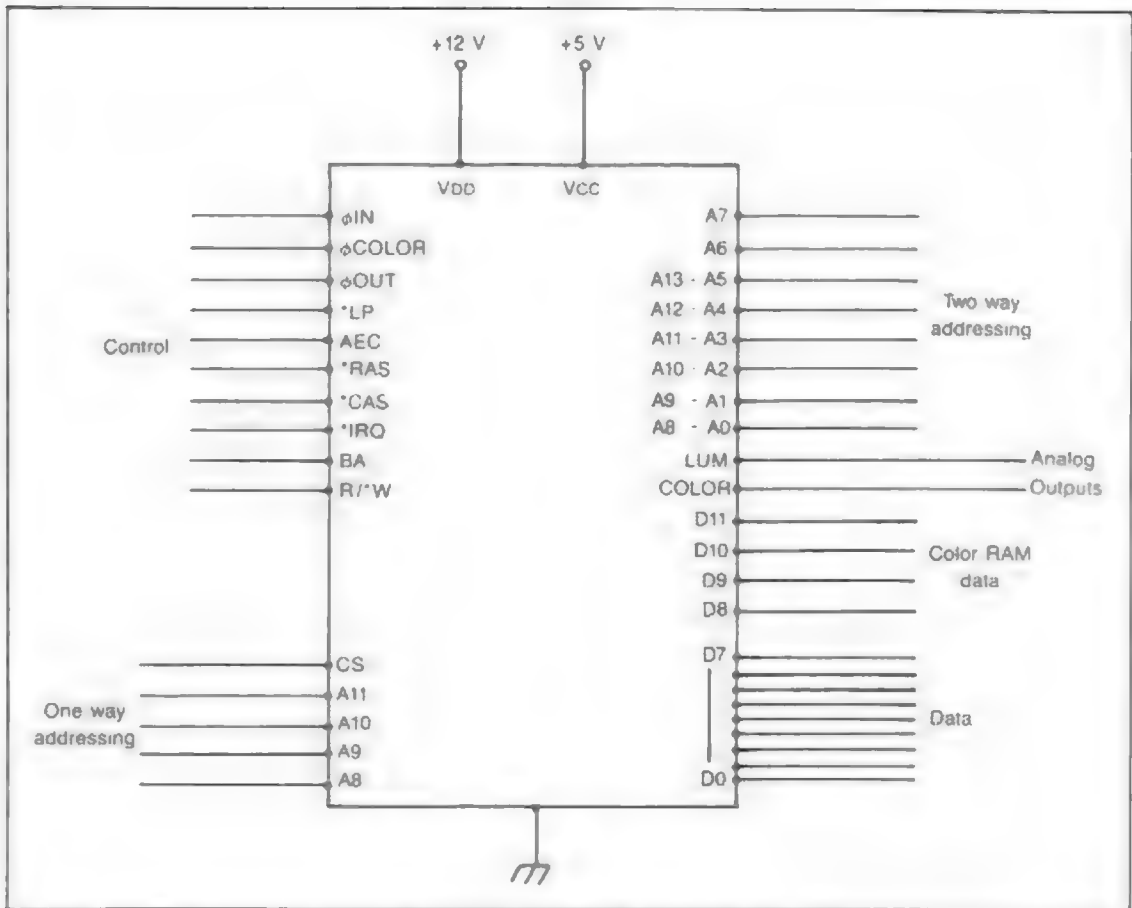


Fig. 5-11 The 48-pin VIC in the C128 is an upgrade of the 40-pin VIC in the C64

luminance or brightness. The four signals are sent to some buffering circuitry in a smaller chip and are then sent directly to the RGBI output port. The important difference between this video output and the VIC's output is the number of columns in the display block. The 8563's output has 80 columns with 25 rows totaling 2000 character blocks on the TV screen.

The VIC chip contains all the capabilities of its predecessor 6567 VIC chip used in the C64. This allows it to operate when the C128 is in the C64 mode. These capabilities include sprites and high-resolution bit-mapped graphics. In addition, the newer 8564 VIC has some extra features. These will be described in detail in Chapter 20. The features

include extended keyboard scanning to handle the extra control lines in the C128 keyboard, an ability to run at a faster 2 MHz speed rather than only 1 MHz, and other things. VIC is the chip that generates the clocks for the C128 in the same manner as it did for the C64. Additional clocks are needed in the C128, and VIC handles this. A 1 MHz clock for the bus and I/O operations, and a 2 MHz clock to regulate some operations (as well as the processor when it is in a 2 MHz mode) are in the VIC. A 4 MHz clock to take care of the Z80 CP/M program running is also in the VIC.

VIC has a number of registers whereby all of its operations can be controlled by writing to the registers and changing register contents. Inciden-

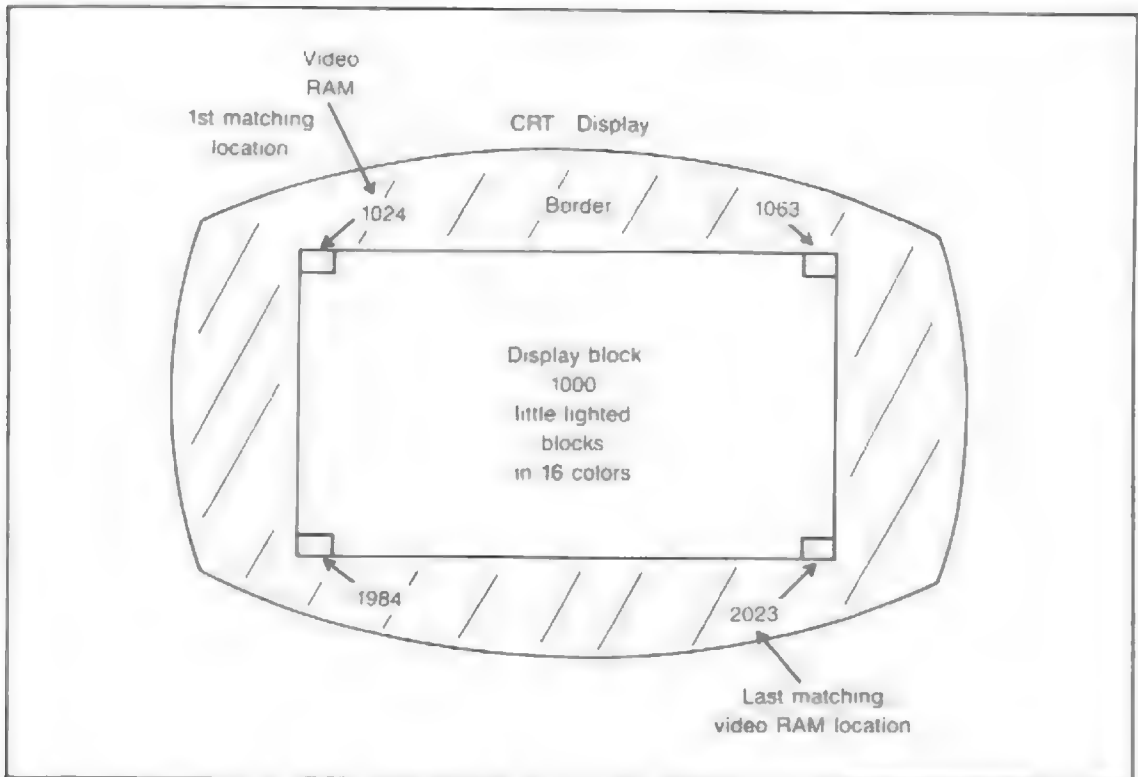


Fig. 5-12. VIC controls the TV display. It generates the video to show 1000 little character blocks. In dynamic RAM there are 1000 eight-bit locations assigned to store code for the characters in these blocks. Also, 1000 four bit locations are assigned in the color RAM to determine the colors of the blocks. Each block needs 12 bits of code to produce a character in color.

tally, VIC can be tested for some defects by reading and writing appropriate bits for the registers. Sample techniques of this sort are found in Chapter 20.

The 8563 Video Controller is a text display chip. The 80-column format it provides is very useful for word processing, spreadsheet analysis and the like. The 80-column mode is used more to produce text than graphics.

The 8563 has two external registers and 37 internal registers. The externals are called the *address register* and the *data register*. By writing to the address register, and reading and writing to the data register, you can access the 37 internal registers.

The 37 internal registers are used to set up the mode of the registers. For instance, in the U.S. the video standard is called NTSC. In Europe the sys-

tem standard is called PAL. The set-up registers set up the C128 to use the desired standard or other standards.

The display registers place and move the text lettering on the TV screen. These 37 registers as well as other details are covered in Chapter 21.

The 8563 chip sends and receives a lot of signals. First of all, the 8563 must keep in constant communication with the MPU with which it is operating. There are nine separate types of signals that it uses—including bits on the data bus, chip selects, register select, R/W and others. Five types of signals let the 8563 work with its memory chips. Then there are five types of signals that the chip uses to get its 80-column display up on the TV screen. These include the video dot clock, the character clock, horizontal sync and the RGBI output. The

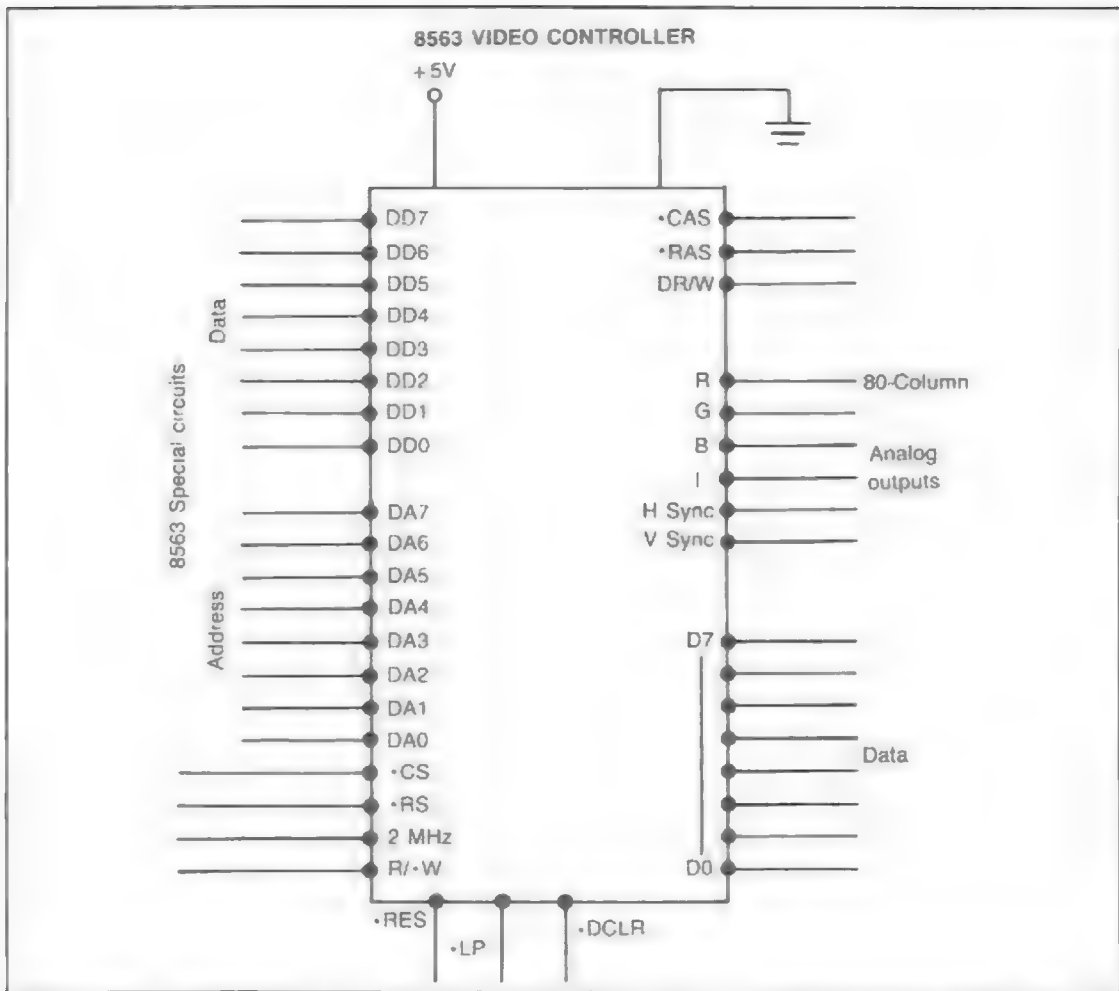


Fig. 5-13. The 8563 Video Controller generates 2000 little character blocks in the display. It shows 25 rows like VIC, but 80 columns. VIC only shows 40 columns.

8563 register map and all of these signals are discussed in Chapter 21. They are important test items when an 8563 is suspected of causing trouble.

THE 6581 SOUND INTERFACE DEVICE

The SID chip is the 28-pin chip found just to the right of the two video output chips in the metal enclosure. SID is a companion chip to VIC. Both types are used in arcade/home video games. While VIC puts graphics on the screen, SID provides the sound effects. If you are using a monitor that does not have

any audio output, then the SID would be useless to you unless you hook up a separate audio arrangement.

The SID, shown in Fig. 5-14, is a sound effects generator. It is designed to be driven by circuits in the 6502 MPU family of which the C128's 8502 is a member. Note that its number is 6581. The 65 indicates the family.

The SID is an advanced computer music synthesizer and sound effects chip. If you are musically knowledgeable, you'll recognize the following qual-

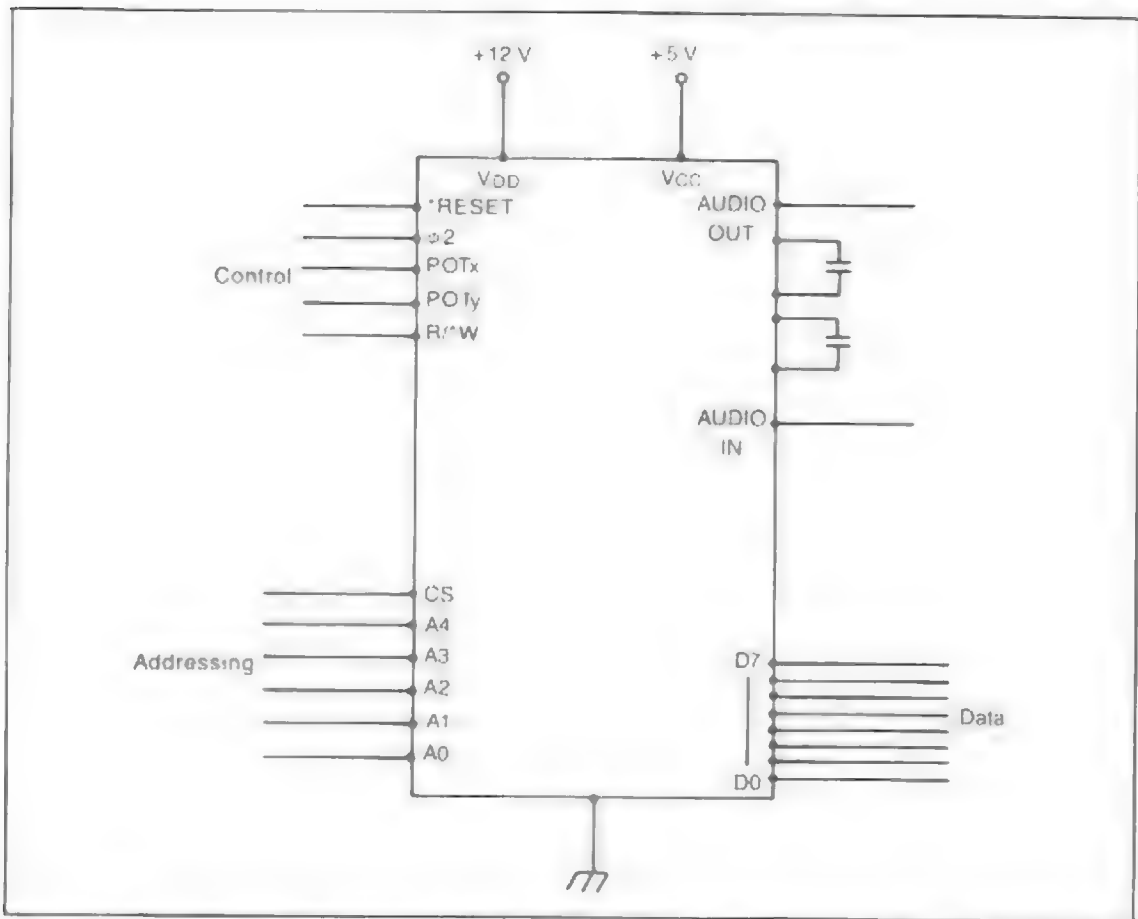


Fig. 5-14. SID is a sound effects generator with address and data lines, control lines and audio input and output.

ities. First of all, the SID is able to provide a wide ranged high-resolution control of pitch. *Pitch* is produced by controlling the frequency of an audio output. There are three audio oscillators in the SID. They are called *voices*. Each voice can be used by itself or in combination with the others as a trio. By controlling the frequency of the oscillators, you can control the pitch of each voice.

In addition to pitch control, the SID permits control of tone color. This is accomplished because the oscillators produce four waveforms at the tuned frequency. Each waveform has its own distinctive harmonic content. By judicious choosing of the individual waveforms, you can control the tone color of the sound.

Besides controlling pitch and tone color, you can instruct the SID's volume level. Each voice has what is called an *envelope generator circuit*. When the program instructs it to, the envelope generator creates an envelope waveform that controls the amplitude modulator. All this creates the desired amount of volume. Lastly, there are programmable filter circuits that further generate complicated, interesting tone colors.

All of these functions can be programmed directly into the SID. Programming these functions also is an excellent test technique to determine which SID systems are working and which ones are not. The electronics and test programming are covered in Chapter 22—a full chapter on SID. You'll

learn about the 29 eight-bit control registers that produce the sound effects for the C128.

THE PLA AND MMU

The 8721 Programmed Logic Array chip, near bottom center on the printboard, and the 8722 Memory Management Unit chip, near right center, are both 48-pin DIPs. They address the locations of the memory map. Both the 8502 and the Z80 are eight-bit MPUs. As is customary, each one sends 16 address bits. Sixteen address bits are only capable of dialing up 64K locations. In the C128, there are 128K RAM locations alone. Then there are 32K of BASIC ROM, 8K of Kernel operating system ROM, 4K of screen editor ROM, 4K of character set ROM, a possible extra 32K of internal ROM, another 32K of external ROM, plus a number of additional register addresses in assorted chips. If the MPU can only address 64K, then how are the rest of the addresses contacted? With the help of the PLA and MMU.

The PLA performs a chip select function. For example, there are four large ROM chips in the C128 plus a fifth empty socket that one more ROM can be plugged into. All five ROMs are wired up in the same way. They are all connected to the same MPU address lines. These lines are able to address all the locations in each chip. If there are 32K locations on

a chip then all of the chips will have the same address numbers. If a number is dialed up then all five chips will be contacted at the same time. In order to pick out only one chip to address, that chip will have to be selected. This is where the PLA comes into the picture.

The PLA receives address lines A15-A10. From the bits on these lines, the PLA forms a chip-select signal. A line connects the PLA to each and every chip that must be selected. Over the correct line goes the chip-select bit that the PLA generated from the A15-A10 address lines. The chip selected turns on while the rest of the chips stay off.

Besides selecting a ROM among the five main ROM circuits, the PLA selects VIC, the color RAM chip, the character RAM, and other things. The details on the PLA will be covered in Chapter 14.

The MMU works along with the PLA. While the PLA handles the chip selections, the MMU controls the management of all the different modes in the computer. The MMU can be programmed. Therefore, in addition to being hooked to the address bus, the MMU is also connected to the data bus, seen in Fig. 15-7.

In the MMU chip there are a series of programmable I/O type registers. These registers have their own addresses on the C128 system memory map.

Table 5-1. The MMU selects the bank number to be used at any particular program line. It chooses from among the chips that are residents of the memory map.

Bank Number	Bank contents
0	RAM Set 0
1	RAM Set 1
2	RAM Set 2 (not in C128, uses RAM Set 0)
3	RAM Set 3 (not in C128, uses RAM Set 1)
4	RAM Set 0, I/O, Functional ROM (empty socket)
5	RAM Set 1, I/O, Functional ROM (empty socket)
6	RAM Set 2, I/O, Functional ROM (empty socket)
7	RAM Set 3, I/O, Functional ROM (empty socket)
8	RAM Set 0, I/O, Cartridge ROM
9	RAM Set 1, I/O, Cartridge ROM
10	RAM Set 2, I/O, Cartridge ROM
11	RAM Set 3, I/O, Cartridge ROM
12	RAM Set 0, I/O, Kernel ROM, Functional ROM Low
13	RAM Set 0, I/O, Kernel ROM, Cartridge ROM Low
14	RAM Set 0, Kernel ROM, BASIC ROM, Character ROM
15	RAM Set 0, I/O, Kernel ROM, BASIC ROM

On the C64 memory map the MMU does not exist. The 8502 in the C64 mode acts just like a C64. In the C64 there is no need for the MMU. The 16 address lines of the 8502 with the aid of the PLA can handle all necessary addressing duties. Chapter 15 goes into detail on all these registers. For now, let's take a quick look at the MMU in the C128 mode.

When the computer is in C128 mode, the default memory arrangement set up by the MMU registers lists 16 banks. Each bank uses 64K of available address space. That way the 8502 or Z80 is able to address whatever bank is in use at that instant. Table 5-1 shows the 16 banks and what combinations of RAM and ROM are assigned to each bank. An interesting note is, banks 0-3 are each able to address 64K. That means the C128 is actually able to address 256K. However there is only 128K of RAM in the machine. Perhaps Commodore has some future plans for the additional 128K of addressing potential. I am sure that somebody, maybe a

third party supplier, will come up with a use for this unused potential.

The MMU is able to allow the 16-bit address bus to address a lot more than the 64K it can only address without help. When you write a BASIC program there is a BANK command that lets you switch from one bank to another with a BANK n, where n is a digit between 0 and 15.

The MMU, besides having these I/O address control registers also has lines that can detect if a C64 cartridge is in place, if a fast serial disk is going to be used, and whether the 40/80 key is up or down. In addition, the MMU has the responsibility of controlling which MPU, the 8502 or the Z80, is in control of the computer.

The MMU is a main chip in the C128. It gives the eight-bit MPUs addressing abilities only found in MPUs with many more address lines. It can also step out of the way and let the C128 act like a C64.

6. Dynamic Random Access Memory

In the lower left-hand corner of the printboard are two sets, eight to a set, of Dynamic Random Access Memory chips, called DRAMs. The generic name for each chip is 4164. Each set contains 64K of DRAM. They total 128K of DRAM. They are the reason the computer is called a C128. As mentioned, the MMU is able to address 256K of DRAM. If an upgrade is made in the future, then this computer's name could be changed to the C256.

Three more RAM chips in the machine are not connected with this group of 128K. Two 4416 chips are under the metal video enclosure, near the 8563 Video Controller. These two DRAMs are not in the C128 or C64 memory maps. They can only be accessed by going through the 8563. They operate with the 8563 and are an important part of the 8563 operation. They will be discussed in Chapter 21 along with the 8563.

One more RAM chip is in the C128. It is a 2016 and is found near the bottom of the board just to the left of the PLA. The 2016 is the Color RAM chip

and works closely with VIC. It will be covered later in this chapter.

Let's examine the DRAM in Fig. 6-1. It is a 16-pin MOS chip. It contains 65,536 individual bit holders, as illustrated in Fig. 6-2. The 65,536 is rounded off and called 64K. Note that there are 64K single bit holders on the chip. It takes a set of eight of them to supply 64K bytes of memory.

To get specific, in computer chips a "K," while loosely thought of as a thousand, is actually 1024. Therefore, if you multiply 1024 times 64, you get 65,536. The reason 1024 is chosen is because computers do all their figuring in powers of 2. The closest binary multiple to 1000 is 2 to the 10th power. Counting this way the numbers ascend thus: 1,2,4,8,16,32,64,128,256,512, and 1024, which is 2 to the 10th power. If you continue the progression, you'll encounter more familiar computing numbers, 2048 (2K), 4096 (4K), 8192 (8K), 16384 (16K), 32768 (32K) and 65536 (64K). Table 6-1 summarizes all these numbers for you. It is a good

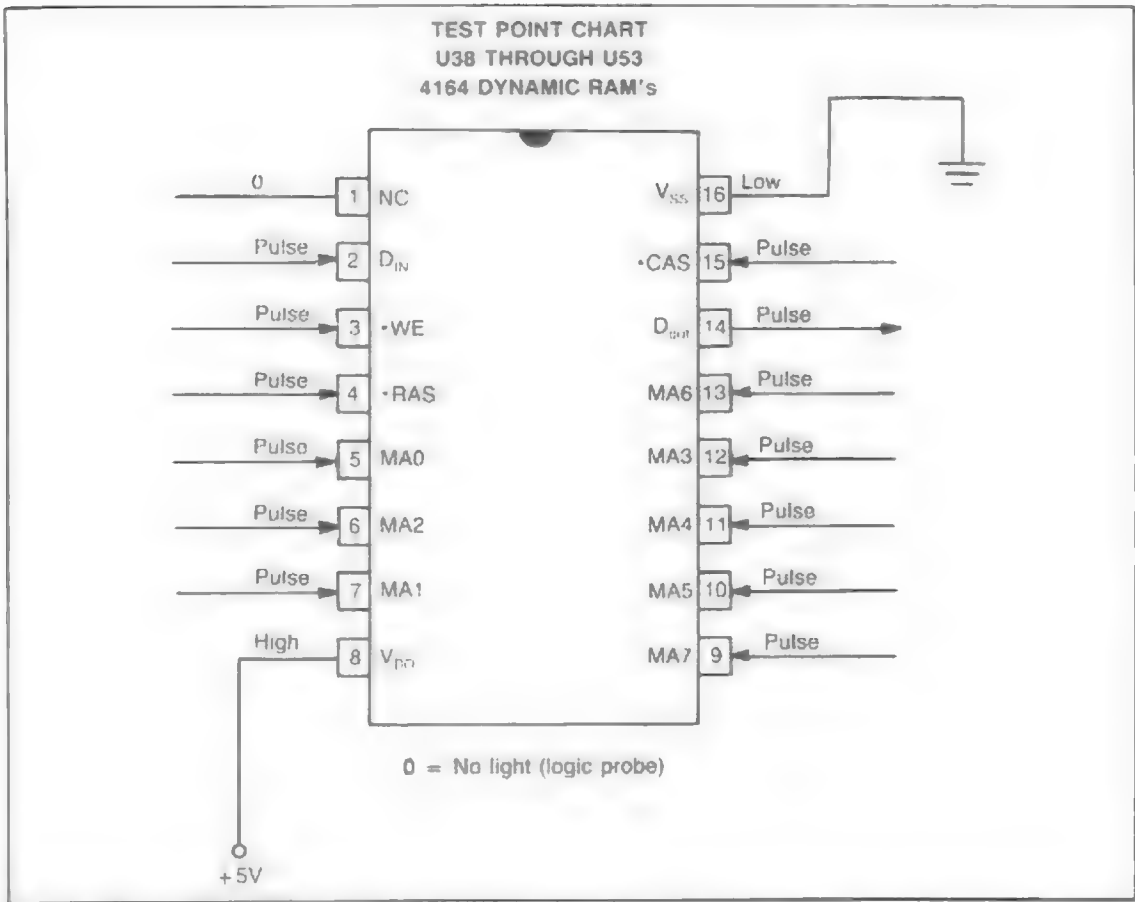


Fig. 6-1 U38 through U53 are the Dynamic RAM chips holding the 128K of memory. The chips are known as DRAMs. This illustration is a Test Point Chart. You can test these pins one at a time with a logic probe. All of the chips have the same logic states on their respective pins.

idea to remember these landmark numbers up to 64K, and even above, as you could find them very useful during troubleshooting and repair.

The term RAM, for *Random Access Memory*, is a holdover from before the microcomputer. The D in DRAM simply means dynamic. The RAM part is not really an accurate description. The way computer professionals view RAM is read/write memory in contrast to ROM which is read only memory. ROMs are covered in the next chapter.

A RAM chip is a little warehouse where you can store bits. It works with other active chips and provides a place where the chips can place bits aside and then recall them. When a busy chip stores bits

it does so by writing the bits over to the storage area. When the chip recalls bits it does so by reading the locations where the bits are waiting.

Two main types of RAM exist: the static type and the dynamic type, called DRAM. In the C128, there is one static RAM chip, U19—the 2016 color RAM. All of the rest of the RAMs are DRAMs: the 16 4164's in the 128K group and the two 4416's, U25 and U23, that provide 16K of DRAM for the Video Controller.

STATIC RAM

In the C64 the VIC works with a static RAM chip, a 2114 that is used to store colors, as shown

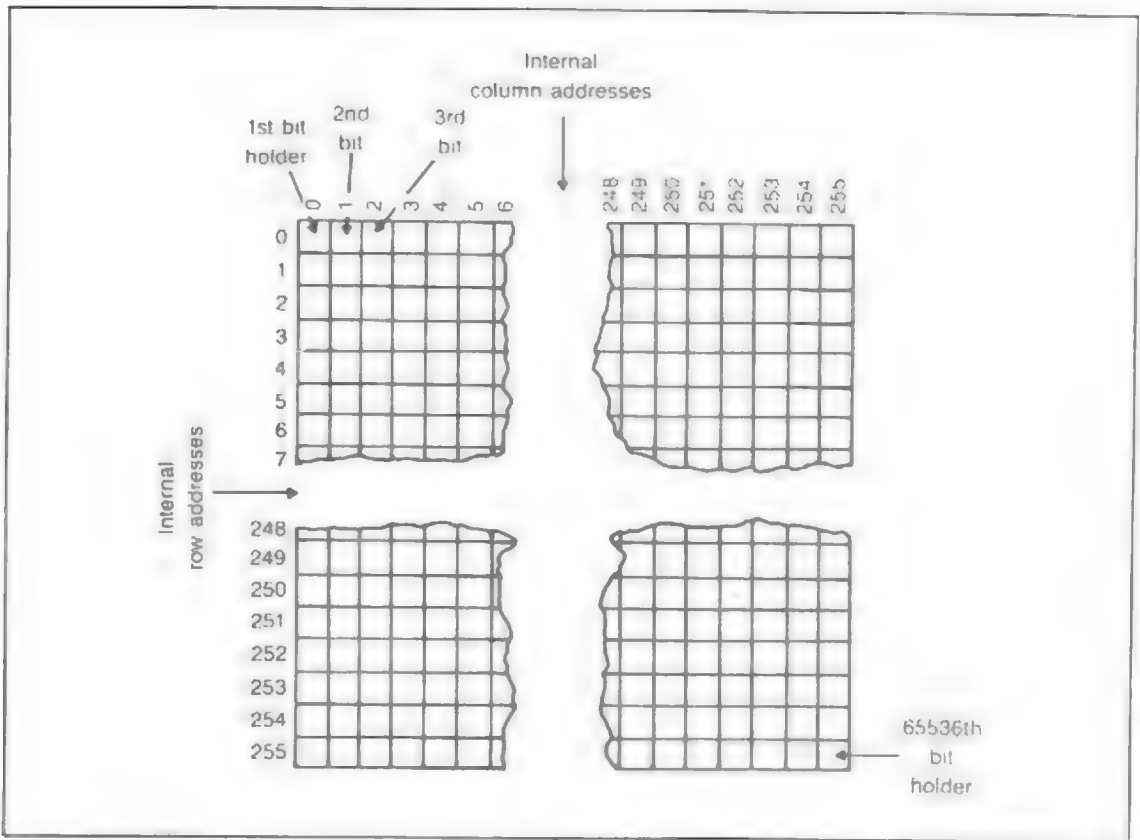


Fig. 6-2. These DRAMs are organized in one-bit registers with 256 columns and 256 rows. This totals 65,536 individual registers on a chip

in Fig. 6-3. In the C128, the newer and larger VIC also works with a color RAM chip: U19 a 2016. The way that the VIC and the 2016 operate together is discussed in Chapter 20. The two color RAMs, although quite different in makeup, operate in much the same way.

The 2114 is an 18-pin DIP. The 2016 is a 24-pin DIP. The 2114 has the bit holders organized in a 1024 x 4 format. Therefore, 1024 individual locations are built into the chip. The 4 means that four bit holders are at each location. Four bits is half a byte and is accordingly called a *nybble*. Figure 6-4 illustrates the arrangement as a tall, skinny four-bit wide structure. Each nybble location has an internal address. The 1024 addresses are numbered 0 through 1023. Zero is the first address and 1023 is the last.

Each bit holder (four to an address) is able to store a high voltage like +5 volts or a low voltage, like zero volts. The highs and lows are code for 1's and 0's. The highs and lows are stored in flip-flop circuits. Each bit holder is a flip-flop circuit in a static RAM. Static RAM uses flip-flops, but DRAMs use other circuits that we will get to later in the chapter.

This tall skinny memory layout is called the memory matrix. Each location is called a register. In this 2114 the registers are four bits wide. The address bus is connected to the rows of addresses from 0 to 1023. The data bus is connected to the four bit holders. The bit holders could be called D3, D2, D1 and D0. All of the 1024 locations have their D3 flip-flops connected to the D3 data bus line. The D2 flip-flops are all connected to the D2 data bus line, etc. The signal on the address bus will deter-

Table 6-1. Good Numbers to Remember.

Powers of 2	Decimal Count	Bits Required
2	0-1	1
4	0-3	2
8	0-7	3
16	0-15	4 (nybble)
32	0-31	5
64	0-63	6
128	0-127	7
256	0-255	8 (byte)
512	0-511	9
1024(1K)	0-1023	10
2048(2K)	0-2047	11
4096(4K)	0-4095	12
8192(8K)	0-8191	13
16384(16K)	0-16383	14
32768(32K)	0-32767	15
65536(64K)	0-65535	16 (2 bytes)

mine which row of four flip-flops can place their data on the data bus. Figure 6-5 illustrates this idea.

When one of the registers is addressed, then it activates the connections between the data bus lines and its bit holders. A copy of the highs and lows in the bit holders will pass over the data lines.

The 2114 in the C64 has ten address lines A9-A0. Ten address lines are able to bring 1024 possible combinations of highs and lows to the chip. Since there are exactly 1024 registers in the 2114, the ten lines are able to bring a separate address for each location in the chip. That is how the memory matrix of the 2114 can be addressed in the C64.

In the C128, the same type of color RAM chip is needed, except that there are two banks of 64K in the 128K of DRAM. The color RAM still only needs four bit holders to a register, but it requires 1024 of color RAM for each bank. The number of

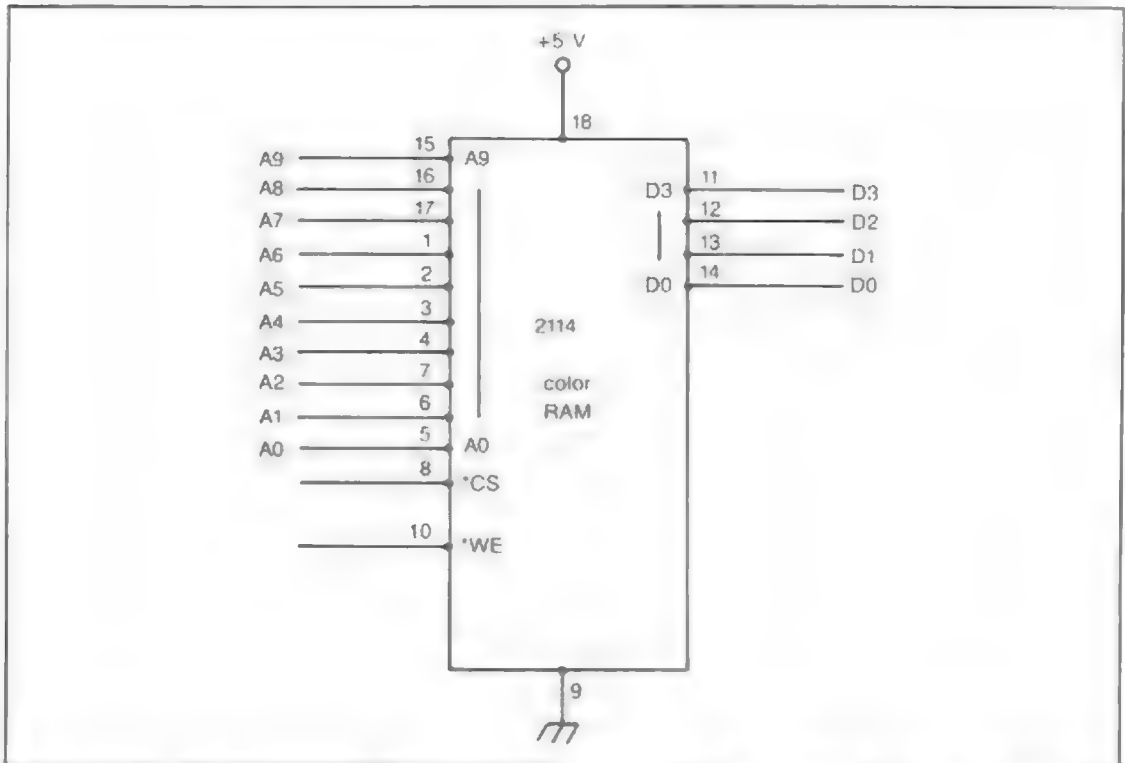


Fig. 6-3. The C64 works with a 2114 static RAM chip. Note it only has four data bus lines, D3-D0. This is a color RAM. It only has four bits in a register. This can produce 16 different codes to generate 16 character colors.

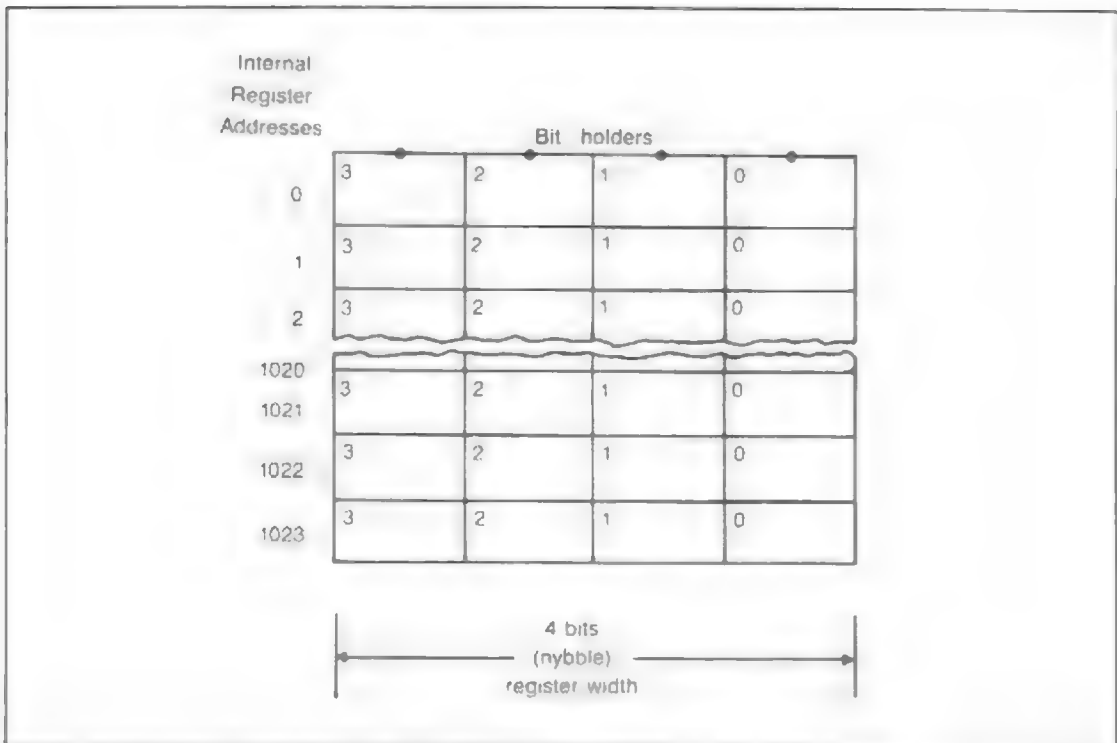


Fig. 6-4 The 2114 is organized in four-bit nybbles. The internal addresses of the nybbles range from 0-1023.

four-bit holders must be doubled. Therefore a new static RAM chip has been employed with 2048 four-bit registers.

Figure 6-6 is the schematic arrangement of pins on the chip. It is a 24-pin DIP. The memory matrix is in a 2048 × 8 format. Note pins 9, 10, 11, 13, 14, 15, 16 and 17 are called data bus lines D0 through D7. Yes, there are eight-bit holders to a register. Only four-bit holders are needed. The rest of them will only get in the way. They must be disabled. Therefore lines D4, D5, D6 and D7 are all connected to 3.3K resistors that go to +5 volts. Since +5 volts is a high, all four unwanted bit holders are said to be held high and inactive. This is one design trick to disable unwanted pins on a chip so they won't interfere with operations.

Because there is the need to address 2048 individual locations, 11 address lines are connected: A10 through A0. Each additional address bit doubles the number of locations that can be addressed.

The 11 address lines are connected to pins 19, 22, 23, 1, 2, 3, 4, 5, 6, 7 and 8.

Pins 20 and 18 are called *CS. The CS stands for chip select. The asterisk in front of the CS means that the pins are held high at +5 volts most of the time. When the high is on the pins, then the chip is turned off. If you desire to select the chip and turn it on, then a low from the PLA is sent to the pins. Note that they are tied together so that one low, called "color RAM," from the PLA activates both pins. They both must be so enabled for the chip to turn on. When the chip is selected it will respond to the 11 address lines.

The asterisk, or on other schematics a line above the CS, is the sign that the chip will be enabled by a low. If the asterisk is not there, it means that the CS pin is being held low while off and will be enabled when a high arrives.

How does the chip know whether to output the contents of the addressed register or input a new

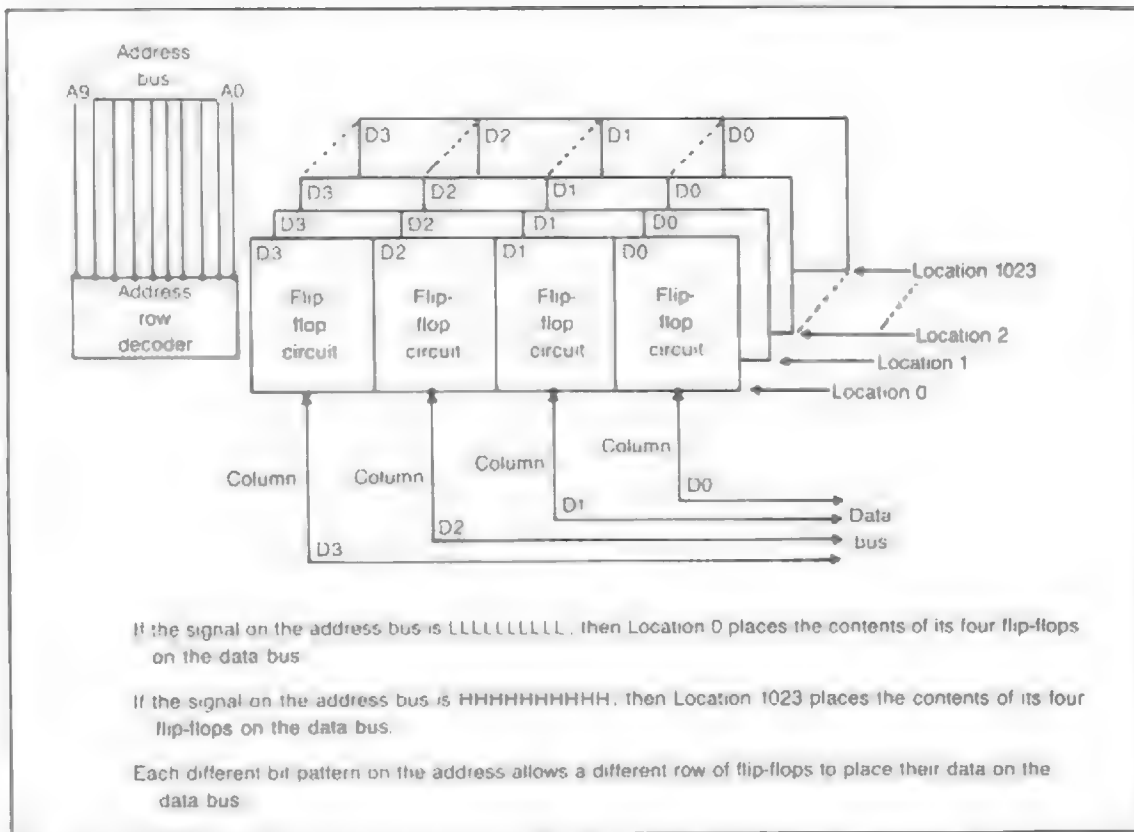


Fig 6-5 Each register in the 2114 is able to store a high or low voltage. The high is the electronic representation of a 1 and the low indicates a 0. The register addresses are arranged in rows and the data bits are in the columns.

set of highs and lows? Pin 21, *WE, makes that decision. WE means "write enable." The status of the write enable pin determines if the chip is to be read or written to. It is the read/write line for the 2016 chip. The asterisk means that the chip is being held high. This is the standby condition. While in this standby, the registers can be read by another chip. Most of the time, the Color RAM chip acts like a ROM and is read. If it is necessary to write to the chip, a low is sent to the *WE pin and the status changes to write.

The last two pins on the chip are the +5 volt power supply input at pin 24 and the ground connection at pin 12. They can be tested with a vom directly or with a logic probe. The +5 volts will read a HIGH on the probe while the ground connection (0 volts) will read a LOW.

Figure 6-7 is the chip's Test Point Chart. Note that the different pins are all labeled with a logic state. Most of them are Pulses. Other chips in the other charts will also have high and low levels. There are also the input power supply voltages like +5 volts and ground connections. At the top of each chip there is a half moon notch which is the keyway for the chip.

The voltages and logic states shown on the pins are the ones that should be present while the computer is in a C128 40-column mode and idling. If you read the pins one by one with a logic probe or vom, a properly operating chip should show these states or voltages. If you read these pins and one or more of the readings are incorrect, then that could be a clue to any trouble you are looking for. If the readings are all okay, then the chip is probably okay. The

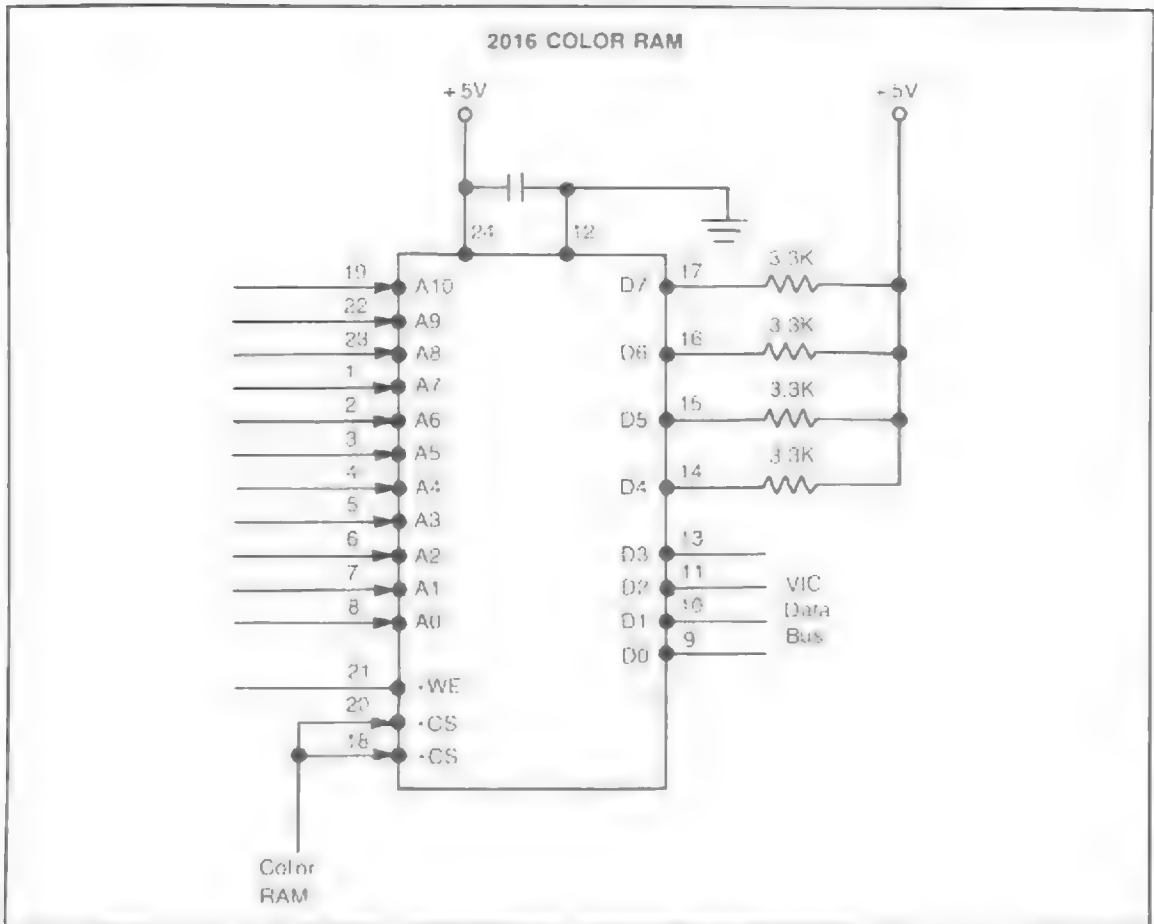


Fig. 6-6. This is the schematic arrangement of the 2016 color RAM in the C128. It has eight data bit connections. Note that four of the data lines are disabled by connecting them to -5 volts. Only D3-D0 are needed to code the character color

use of charts like these are one of the test techniques that professionals use to pinpoint trouble.

The beginning of this color RAM discussion described what each pin on the 2016 chip is doing. As you make logic probe tests on each pin, note carefully if the pin is doing what it is supposed to do. When a pin is found that does not have the correct high, low, pulse or dc voltage, then try to puzzle out the cause. If you solve the puzzle, which is mostly mind work, you will then know what part or connection is defective. Once the diagnosis is made, then the physical repair of replacing a part or repairing a connection or copper etch spot can follow quickly.

DYNAMIC RAM

One of the most striking differences between static and dynamic RAM is the organization of the bit holders. The static 2016 chip is organized in a 2048×8 layout with four of the eight bit holders disabled by holding them high. Figure 6-8 shows a 2048×8 chip layout. In a chip like this one, all of the byte-sized registers are contained on one chip. The layout of the 4164's in the C128 is entirely different.

One dynamic 4164 chip has a layout of $65,536 \times 1$. This means there are 65,536 registers on the chip but each register is only one bit wide. Each register or each bit as it turns out, has its own address.

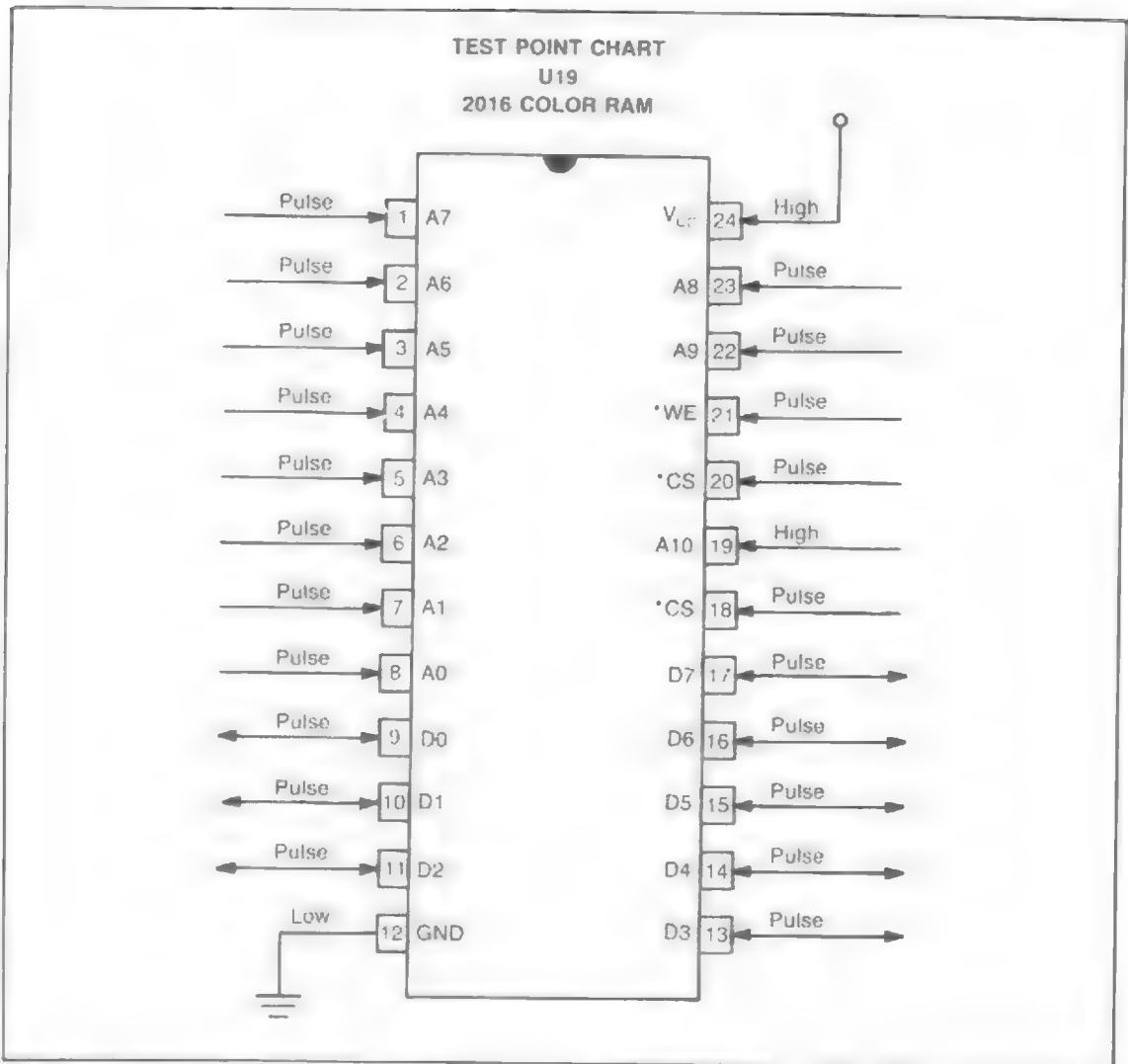


Fig 6-7 This is the Test Point Chart for U19 the 2016 color RAM. It shows the actual pinout, the direction of signal flow and the logic state that should be present while the C128 is idling in C128 mode.

Figure 6-2 showed the layout. You can see all the single bit registers in comparison to the 2016's eight bit registers.

In the C128, the 16 DRAMs are arranged in two banks of eight DRAMs each. Eight DRAMs are wired up as one unit. They are connected in parallel. Figures 6-9, 6-10 and 6-11 describe the layout. All of the eight chips are identical. Each chip has the same number of bit holders. Because there are eight

chips to a set there are enough bit holders on all the chips to form 65,536 bytes. All they need is to be wired correctly. The first thing that is done is to wire all the address lines to the same pins on all chips as in Fig. 6-11. That way, when an address comes over the address bus, all eight chips address the same location number simultaneously.

The next wiring is to connect each chip in the set to a different data bus line. That way, as the

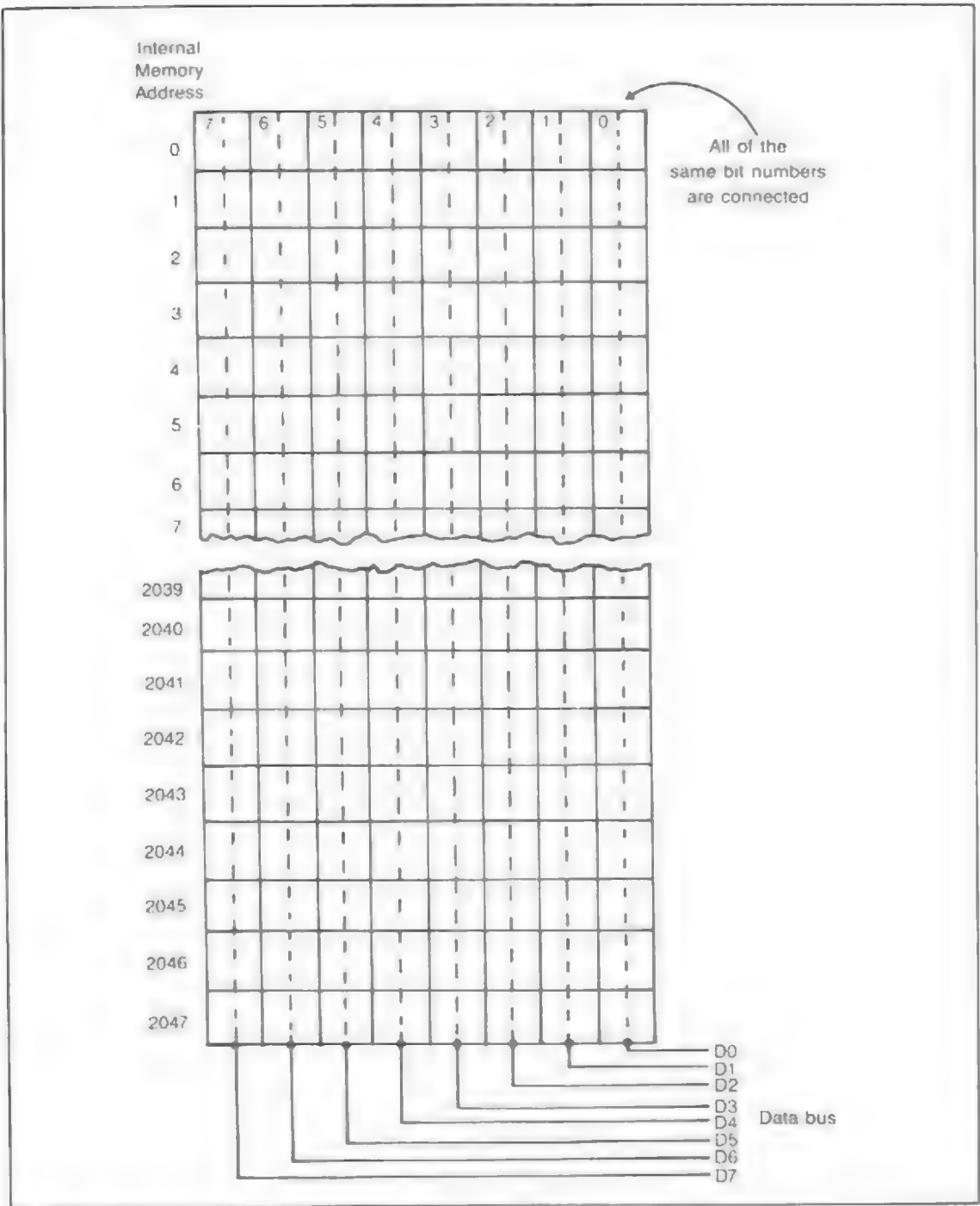


Fig. 6-8. This is a static RAM chip organized in a 2048 x 8 fashion. All of the byte-sized locations are found on the single chip

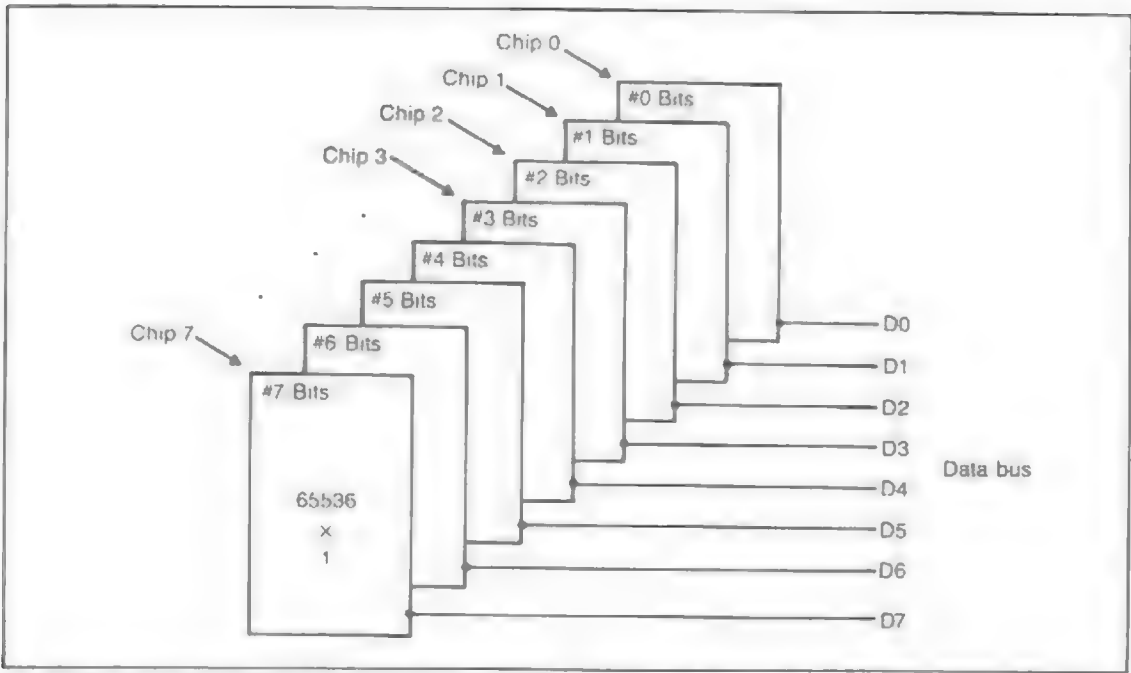


Fig. 6-9. The 4164's in the C128 are organized in a 65,536 \times 1 fashion. It takes eight 4164's to produce byte-sized locations: one bit of a byte on each chip.

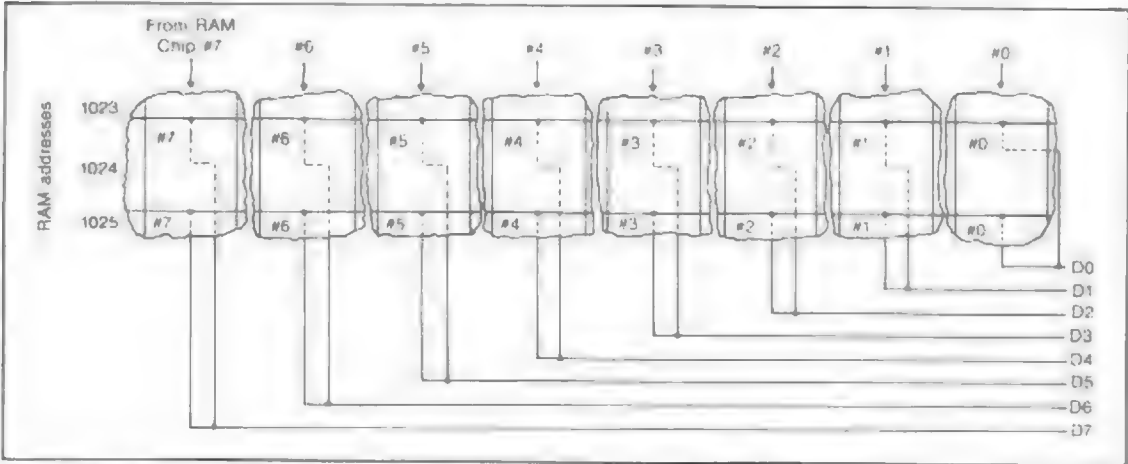


Fig. 6-10. RAM address 1024 is shown as one bit torn out of each RAM chip.

same number location in each chip is addressed, each of the bit holders addressed can either receive a bit or output a bit. Each chip contributes one of the eight bits in the addressed byte. A byte-sized register is found in the set of eight—one bit from each chip

forming the eight-bit register. Figures 6-9 and 6-10 illustrate the design.

The bit holders in the dynamic RAM are not flip-flop circuits as in the static RAM. The bit holders are capacitance that is formed in the MOS gates as

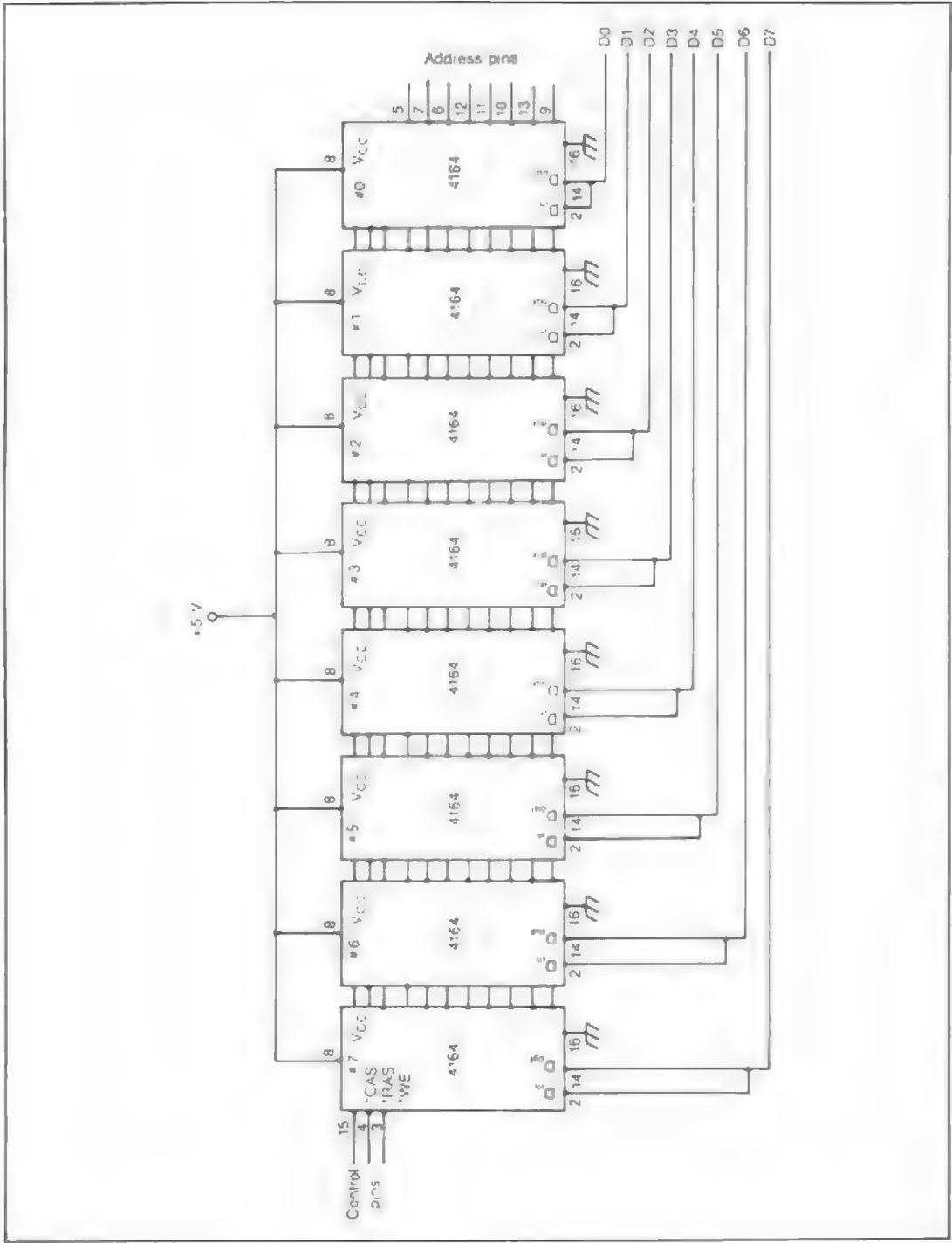


Fig. 6-11. Though only one data bus line goes to each chip, all of the address bus lines go to all of the chips simultaneously.

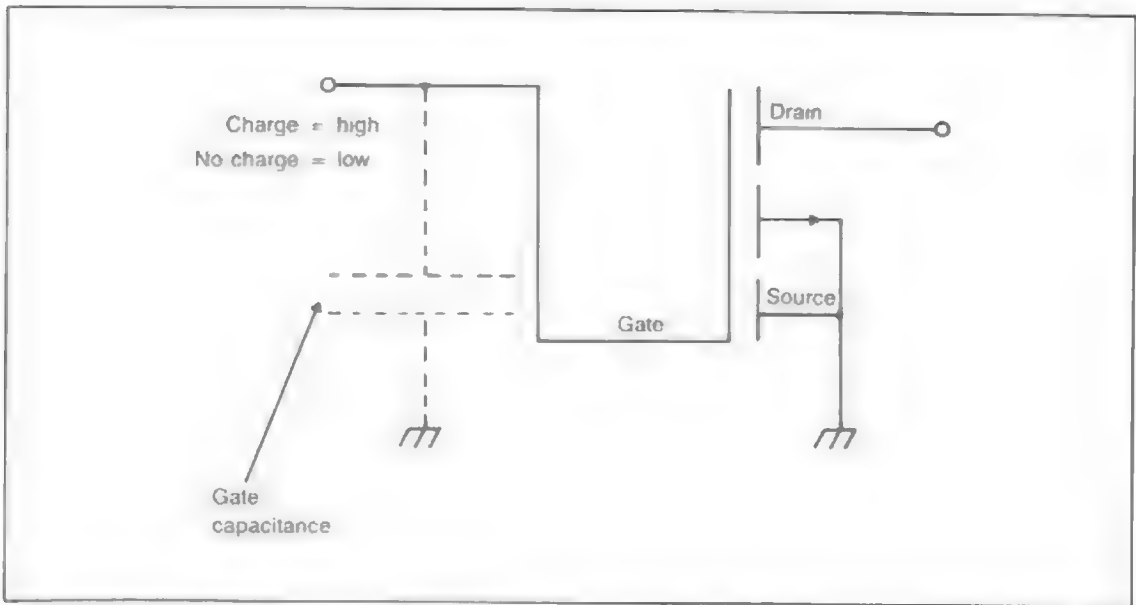


Fig. 6-12. DRAMs do not use flip-flops as bit holders. The gate capacitance of the FETs on the chip hold the highs and lows

shown in Fig. 6-12. In the FETs that make up the chip circuits, the gate is separated from the channel by the sensitive glassy insulator. The gate becomes one lead of a capacitor and the rest of the FET becomes the other lead. The insulator forms the dielectric. In this way a tiny capacitance is formed. The capacitance, like any capacitor can be made to hold a charge.

For the charge to represent code for the binary data, the following convention has been arranged. If the capacitor is charged, it represents a high (binary 1). When there is no charge the capacitor is storing a low (binary 0). The amount of capacitance is tiny but adequate. The memory matrix on the 4164 DRAM is really nothing more than a clever circuit around a grid of tiny capacitors.

Memory Refresh

The 2016 static memory chip has active nybble-sized registers, four active flip-flops in each register. The flip-flops are dc stable. That is why the chip is called static. If you place a flip-flop into a high state or a low state (representing a 1 or 0), the flip-flop will hold that state as long as the power is on. Nothing more has to be done. If you want to read a reg-

ister, then you address it. You can make as many copies of the data in a static register as you want without losing the stored state.

This is not the case with the dynamic RAM. The 4164's are storing their states as capacitance charges, not stable flip-flop transistor conductions. The capacitors contain tiny charges. The capacitors can only hold their charges for a short period of time. To be exact, the capacitor charge will fall to a useless voltage level in a little more than 3.66 milliseconds (thousandths of a second). The capacitors must be recharged, called refreshed, at least once every 3.66 ms.

As Fig. 6-13 shows, the VIC provides the refreshments. Pins 19 and 20, signals *RAS and *CAS, of the VIC are the ones involved with maintaining the capacitance charges. These signals are applied to all the 4164's at the same time. The RAS stands for Row Address Strobe and the CAS for Column Address Strobe. The details on the signals are found later in this chapter. These signals are designed to recharge the grids of capacitor bit holders in each chip's memory matrix.

In any computer using DRAMs, some sort of refreshing system must be used to keep the mem-

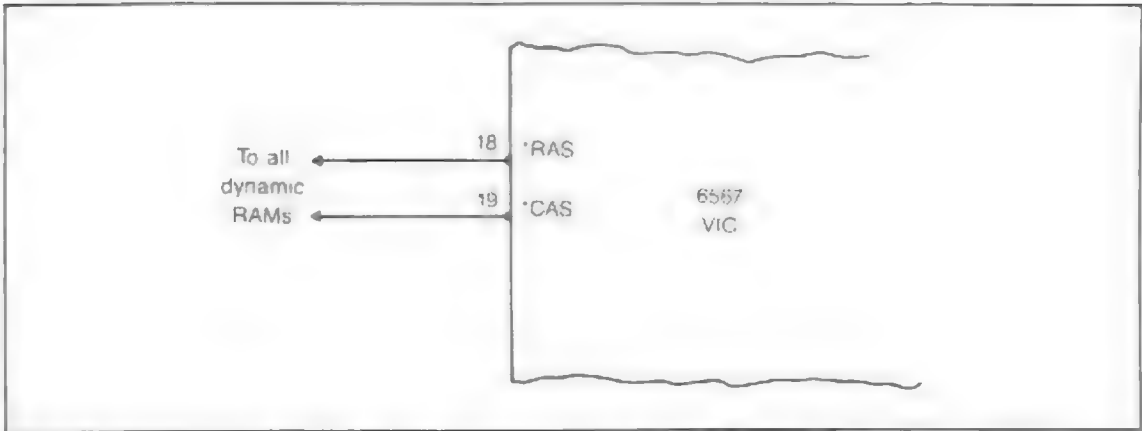


Fig. 6-13. Pins 19 and 20. \cdot RAS and \cdot CAS, deal with constantly recharging the tiny capacitance bit holders in DRAMs.

ory full of data intact. There are all sorts of schemes. The C128 uses special signals generated in VIC to do the job.

Memory Layout

As Fig. 6-2 illustrated, the bit holders are laid out in a grid of 256 rows and 256 columns. Actually, as Fig. 6-14 shows, the layout is divided in half with two grids of 128×256 , with a band of sense amplifiers inbetween. But for all practical purposes, you can think of the grid as 256×256 . Consider the total grid as a single component.

Figure 6-15 shows a functional block diagram of the inside of a 4164 RAM chip. Attached to the

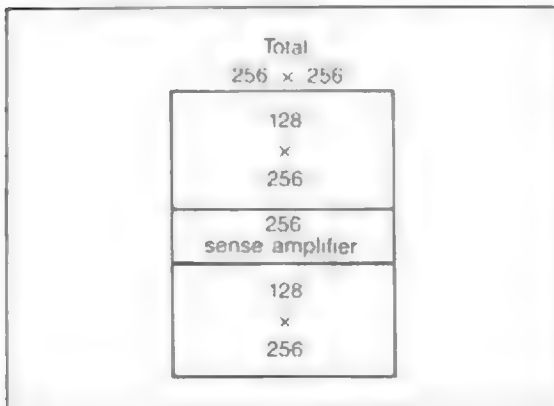


Fig. 6-14. The microscopic insides of the 4164 actually has two grids of 128×256 with 256 sense amplifiers inbetween.

bottom of the grid is a column decoder circuit. It is able to contact every one of the 256 columns. Connected to the side of the grid is a row decoder circuit. It is able to contact every one of the 256 rows.

The column decoder circuit is the data input/output circuit. When a bit in a grid is addressed and becomes activated, the column decoder is there to help out. If the bit is being read, the column decoder receives the high or low contents of the bit holder at one of its connections. When the bit is written to, the column decoder will be the circuit holding the data and will transmit the data to the addressed bit.

The row decoder circuit is part of the bit holder addressing mechanism. The column decoder is also part of the addressing system. The column decoder helps out with the addressing in addition to handling the data in and out.

The dynamic RAM chip is addressed differently than the static RAM. From the mind's eye, a static RAM is a high stack of individual byte locations, and a dynamic RAM chip is a grid of bit-sized locations. Each location has its own address. The 4164 chip has 65,536 locations, but they are arranged in a 256×256 grid. 256×256 equals 65,536.

A single bit is addressed by using two addresses: a column address and a row address. The desired bit holder is the location where the two addresses intersect. Therefore, the address sent out by the MPU over the address bus is in two parts. Eight of the address bits are to locate one row ad-

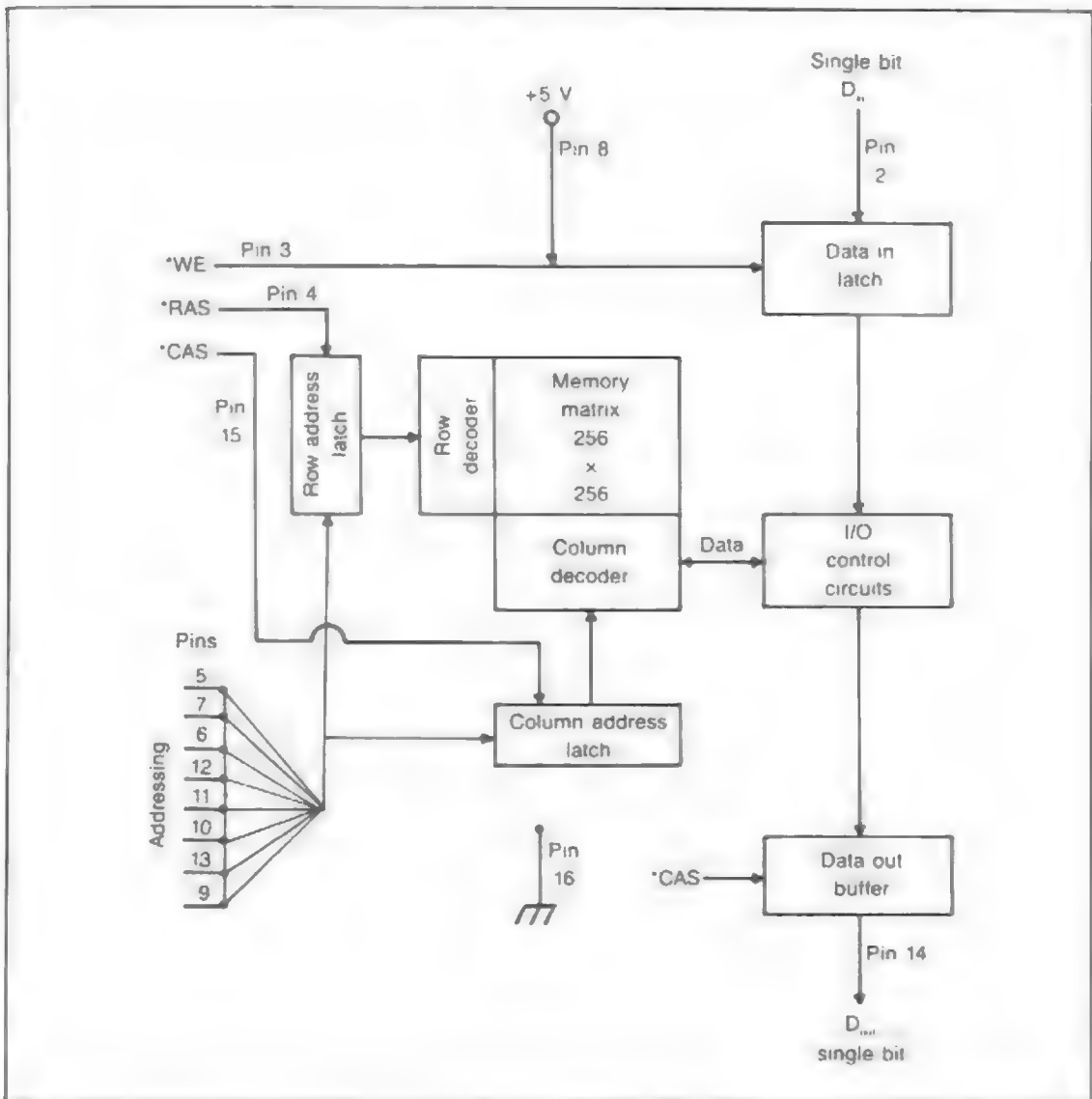


Fig. 6-15. An address in a 4164 is contacted by first addressing one of the 256 rows and then addressing one of the 256 columns. The desired bit is found at the resulting row-column intersection. That bit then opens up and can be read from or written to.

address, and the other eight bits find the column address. The address bits all go to the RAM chips at the same time. The same location bit, on each chip of the RAM set is addressed. The contents of each bit that is addressed this way can then enter or exit each RAM.

Each chip holds one bit of the byte that the MPU wants. All of the number 7 bits, in all of the 64K bytes, are located on chip number 7 of the eight chip set and are attached to data bus line number 7. All of the number 6 bits are on chip number 6 and are attached to data bus number 6, and so on.

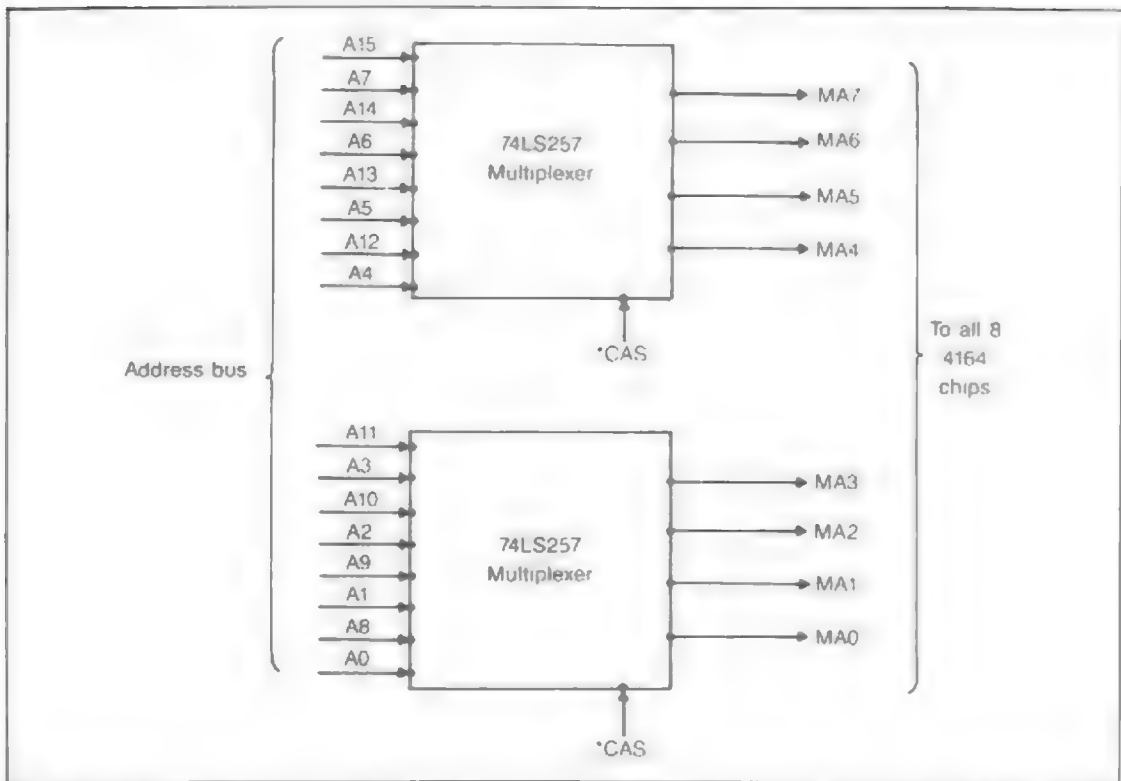


Fig. 6-16 Two multiplex chips perform the DRAM address chores. First they output the row address and then, over the same lines, the column address.

The RAM chips in the C128 are addressed through two 74LS257 multiplex chips, as illustrated in Fig. 6-16. All 16 chips are addressed at the same time. As each location in every RAM chip is addressed simultaneously, two bytes are actually addressed. The PLA is only selecting one bank of eight chips at a time, however, and only one byte of data is accessed. The addressing gets complicated but between the 74LS257 chips doing the row/column addressing and the PLA doing the chip selecting the correct byte of bit holders are contacted. There is more about this addressing in Chapters 12, 13 and 20. The internals of the 74LS257 multiplexers are covered in Chapter 8.

Operation

Figure 6-1 is the Test Point Chart for all 16 4164's, U38 through U53. All 16 chips should read

the same. All 16 pins on each chip will have the same logic states. They are all pulses except the +5 volts power on pin 8 reads high and the ground connection on pin 16 reads low. These states will be present when the C128 is on and idling.

There are eight input lines from the multiplex chips to all of the 4164's. The lines are called MA7 through MA0. The MA stands for multiplexed addresses. The eight lines are derived from the 16 address lines that the MPUs put out. The 16 address lines are connected to the two multiplex chips. Eight bits at a time are needed by the 4164's. Eight bits are needed to address the 256 row addresses. Then eight more bits are needed to address the 256 column addresses. Pins 5, 6, 7, 9, 10, 11, 12 and 13 receive eight bits at a time—first the row address and then the column address. The multiplexers send the bits that way as shown in Fig. 6-16.

At pins 4 and 15, signals from VIC, \bullet RAS and \bullet CAS arrive. These signals, the row address strobe and the column address strobe are there to trigger the two types of addresses into the decoders in sync. As the multiplex chips feed the 4164's first with row addresses and then with column addresses, the \bullet RAS signal strobes the row bits into the row decoder and the \bullet CAS signal strobes the column bits into the column decoder.

This multiplexing plan uses only eight address pins on each of the 4164's. The savings of eight lines allow the 4164 chips to be placed in a 16-pin package.

Figure 6-11 illustrates that the 4164 has two data pins at 14 and 2. Note that the data pins are wired together and act as a single pin. The connected pins attach to their respective eight data bus lines.

The two lines are for both input and output with the data bus, as shown in Fig. 6-15. Internally, the two pins are wired in with the column decoder circuit. The data-in connection is to the data latch. The data latch is a temporary holding register for the incoming data. The data latch is also connected to pin 3, \bullet WE, the write enable pin. During a write operation, \bullet WE goes low which activates the latch and lets the latch transfer the data to the column decoder. The decoder installs the data into the addressed bit holder.

When data is leaving the chip it is passed from the bit holder to a data output buffer stage. The buffers are connected to the \bullet CAS inputs. When \bullet CAS strobes, the data output buffer permits the outgoing data to leave the chip and head out over the data bus.

Timing

The timing diagram in Fig. 6-17 summarizes how the 4164 memory array is controlled by the three signals \bullet RAS, \bullet CAS and \bullet WE. During the addressing as seen on the top waveform, the \bullet RAS strobe signal comes along into pin 4, till the middle of the row time, and then falls. This falling edge strobes the row bits into the row address latch. \bullet RAS then remains low for the remainder of the cycle. Meanwhile \bullet CAS is moving into pin 15. As the column address time arrives \bullet CAS falls. This falling edge strobes the column bits into the column address

latch. \bullet CAS then remains low for the rest of the cycle. The two latches in turn send their bits to their respective decoders. One of the 256 rows is addressed and one of the 256 columns is addressed and their intersection bit is activated.

While the above was going on the \bullet WE signal sets up either a read or write condition for the eight selected bits, D7 through D0, one on each chip in the RAM set. In Fig. 6-17 a write condition is arranged by \bullet WE going low. A low at pin 3, the \bullet WE entranceway, causes the data-in latch to pass data written to the array to be installed in the memory bits. A high at \bullet WE keeps the data-in latch shut and lets the chip output the addressed data to the data bus.

As you can imagine, the timing of the three signals is critical. During routine servicing, however, the timing is not normally a factor requiring testing. The timing is an important consideration during design. The important part that timing plays from a repair point of view is its feature during a chip replacement. You'll see the timing in the spec sheet of a new part referenced as access time. For instance, the 4164 comes in a number of timing versions. The access time on different 4164 chips could be 150 nanoseconds, 200 nanoseconds or 300 nanoseconds. What do these different timings mean?

Different computers run at different speeds. For example, the C128 in the C64 mode runs at a clock rate of 1 MHz. This means that the 8502 in that mode completes one million full cycles in one second. One cycle takes a millionth of a second, or as it is called, a microsecond. In one microsecond there are 1000 nanoseconds. A nanosecond is a billionth of a second. When a 4164 is said to have an access time of 200 nanoseconds, that means it uses 200 ns of the 1000 ns in one clock cycle to be accessed, either written to or read from. The 1000 ns clock cycle comprises one full execution of the MPU.

During the 1000 ns execution cycle, the 8502 must address the 4164 array, send the signals to control it, open up the addressed bits, either read or write to them, and then get ready for the next cycle. A 200 ns array is fairly fast and can easily receive the control signals, open the addressed bits, and complete the read or write. In fact, even the

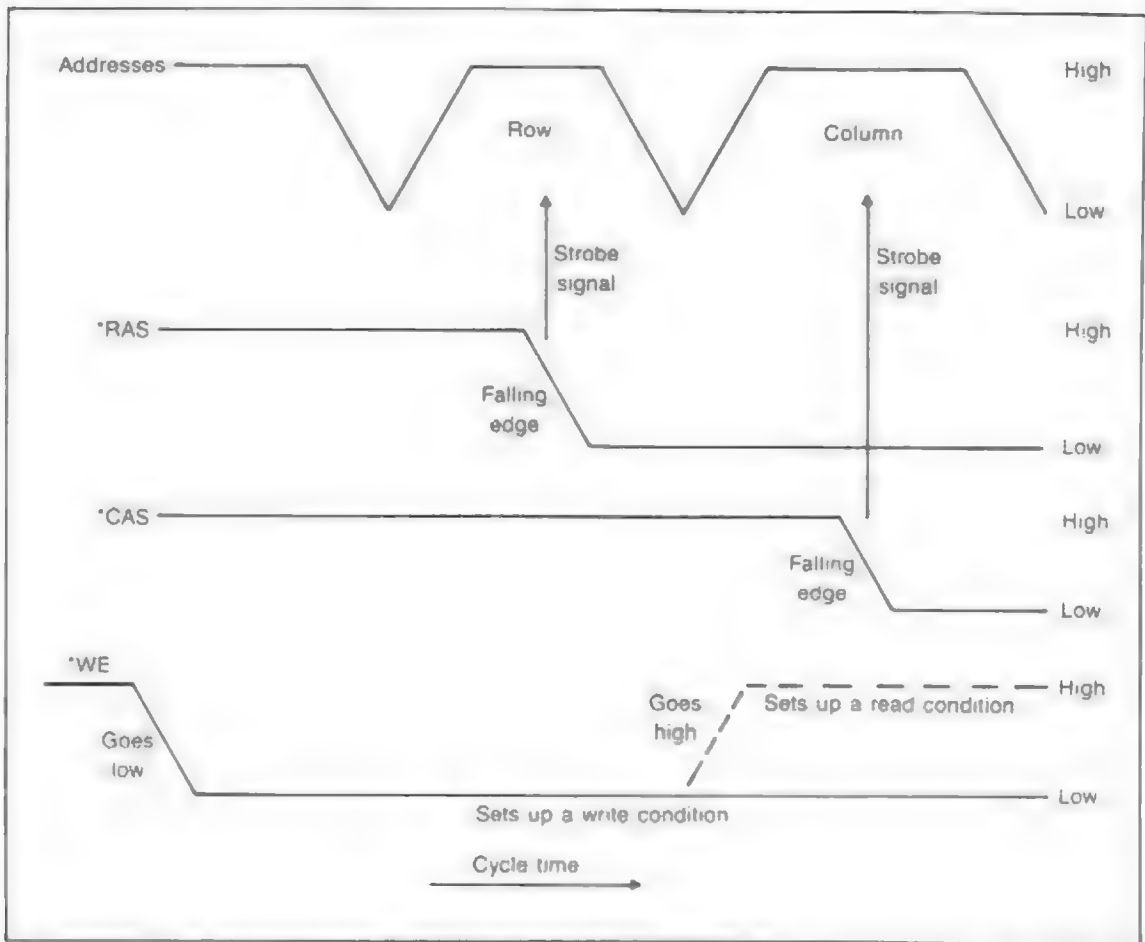


Fig 6-17 The 4164 chips are controlled by the three signals \cdot RAS, \cdot CAS and \cdot WE. During a cycle, the row addresses are strobed and then the column addresses are strobed. During that time \cdot WE sets up a read or write condition.

300 ns chips are easily accessed. If on the other hand, you had some sort of chips that had a 1000 ns access time, you would be hard put to get them fully accessed with a 1 MHz clock. You'd have to get a slower clock or add additional circuitry to gain proper access.

In C128 mode the 8502, besides running at 1 MHz, also can run at 2 MHz. This speedup in frequency cuts the cycle time down to 500 ns. With the 4164 array being able to be accessed in 200 ns, the cycle time is still satisfactory. The action just proceeds in a quicker fashion. The Z80 that conducts the CP/M mode uses the same 4164 array.

Refresh Timing

The 4164's, as DRAMs need to be constantly refreshed. The refreshing at first glance looks impossible. There are 128K locations in the DRAM set that are to be refreshed. That means 131,072 tiny capacitances must be recharged at least once every 3.66 milliseconds or they will lose their information as the charge drops below a certain voltage. Fortunately each bit holder does not have to be individually refreshed. The refreshing can be accomplished by simply addressing rows. When a row is addressed all the bits in the row are recharged.

Also notice that the refresh time is in milli-

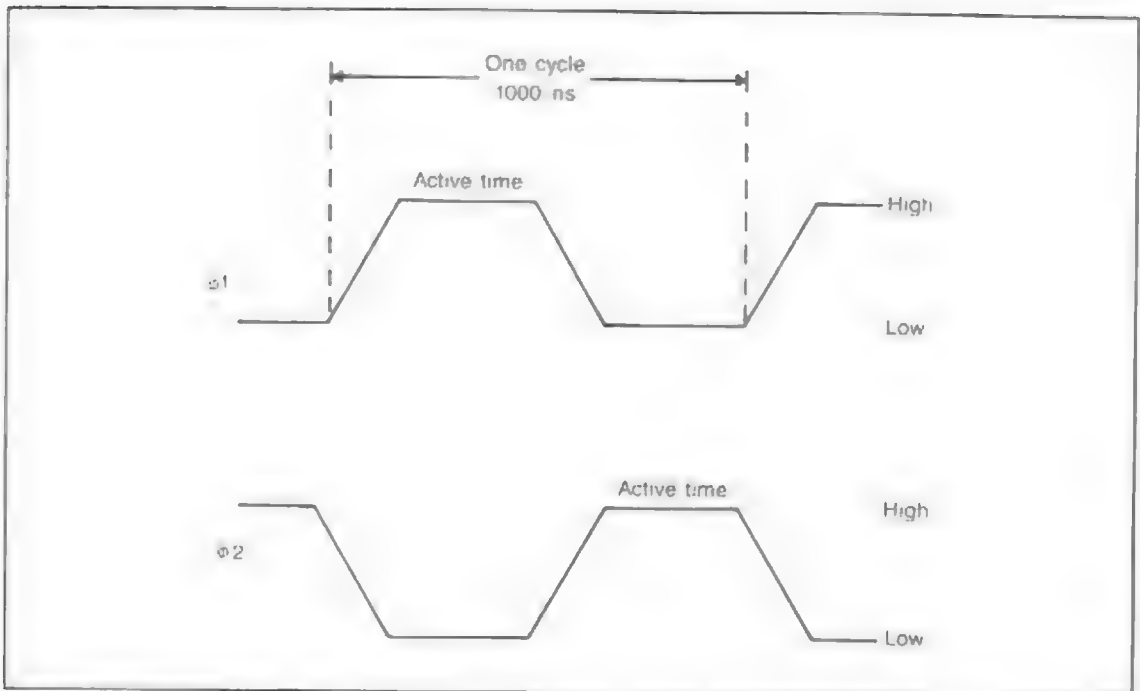


Fig. 6-18. The 1 MHz clock has a 1000 nanosecond cycle time. About half the time $\phi 1$ is active high and the other half, $\phi 2$ is active high. $\phi 1$ deals with the addressing while $\phi 2$ conducts the data movement operation.

seconds, a thousandth of a second in comparison to the micro- and nanosecond that was also mentioned.

Besides the slower timing needed for refreshing, as mentioned it is also not necessary to do anything but address the 256 rows in all the chips simultaneously to recharge the tiny capacitances. The electrical action of the addressing restores the charge in the row that is addressed.

The requirement is that every row must be addressed within the 3.66 millisecond refresh period. At the 1 MHz clock rate it only takes a fraction of the required time to handle the refreshing.

A dynamic RAM refresh controller in the VIC is able to refresh all 256 rows in plenty of time. The refresh circuit uses the ϕ RAS signal to address the rows. It does the job while the computer is busy do-

ing other jobs, not during the time the RAM is being accessed.

The 8502 is controlled by two clock signals shown in Fig. 6-18. They are called phase 1 ($\phi 1$) and phase 2 ($\phi 2$). These will be discussed further in Chapters 12 and 17. $\phi 1$ is a signal that drives the internal circuits of the MPU. It is active about 500 nanoseconds of the 1000 ns total cycle time. $\phi 2$ is a similar signal that is active the other 500 ns of the 1000 ns cycle time. $\phi 2$ drives the rest of the computer external to the 8502. The 4164 array is accessed during $\phi 2$.

During $\phi 1$, while the MPU is busy internally and the 4164 is waiting to be accessed, the VIC goes ahead and refreshes the 256 rows in the array.

7. Read Only Memory

A computer, for the most part, is a hunk of machinery. It needs an operator to get it to turn on and start working. It needs brains to tell it what to do. The brains are in the system ROMs either internal or external. In the C128 most of the ROMs, the so called read only memory, are plugged into sockets on the printboard or plugged into the expansion port as a cartridge. There are five sockets on the board for five 28-pin ROMs, U32, 33, 34, 35 and 36. U's 32, 33, 34 and 35 sockets contain ROM chips. U36 is an empty socket ready for an additional ROM for expansion purposes. In addition there is one 24-pin ROM, U18, the character ROM which is soldered in.

The ROMs themselves are also hardware pieces. Inside the ROMs, though, are human-produced computer programs that run the C128. These programs are loosely referred to as the operating system (OS). In the ROMs are OS's for BASIC and Kernel. In the C64 mode, 16K of ROM is set aside for about 8K of BASIC and 8K of Kernel. For the C128 modes, up to 48K of ROM is avail-

able. The BASIC system of version 7.0 has a high and low section. Between BASIC 7.0 and the Machine Language Monitor, 32K of ROM space is used. With the Kernel, some I/O, and the 40/80 column editor, another 16K of ROM space is utilized. CP/M doesn't have to use much of the ROM operating system. It loads its OS program from the system diskette that comes with the C128.

In the last chapter, it was shown that RAM is a storehouse for binary data bits. The bits can be installed, removed, and reinstalled continually. In fact, the data movement in and out of RAM is the majority of the work that the computer performs. The RAM is both read and write memory. ROM on the other hand is read-only memory. It holds binary data bits engraved in the silicon. The bits are burnt into the silicon in a factory and cannot be removed or replaced. Once a bit is burnt into the chip there is no replacing it. To change the program the ROM must be physically changed.

There are all sorts of schemes to burn programs into ROM chips. Figure 7-1 shows one of the com-

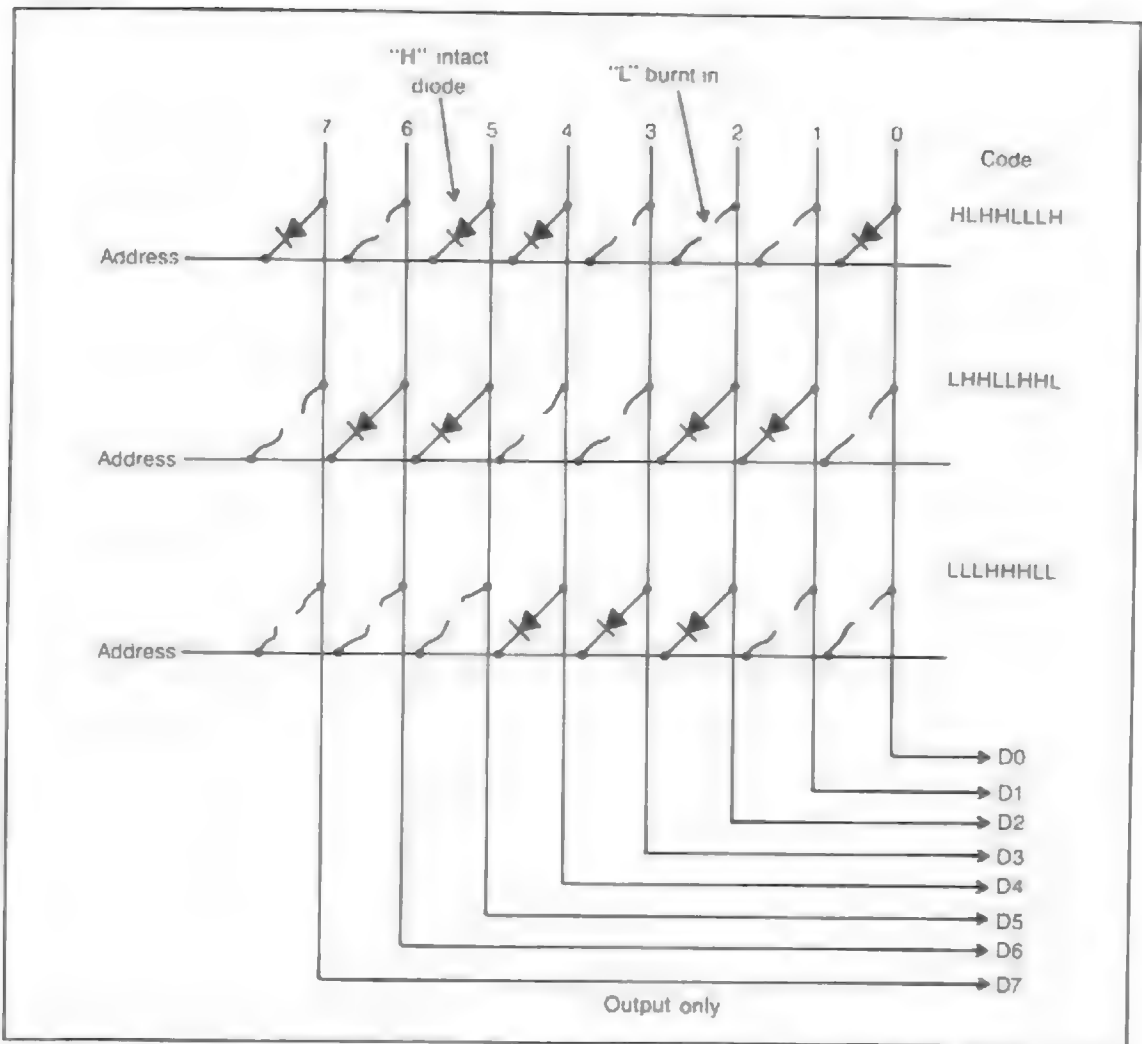


Fig. 7-1 One of the many ways to burn a program into a ROM chip is with diodes. A chip is built with diodes connecting rows and columns. If a diode is left intact it could be code for a high. If the diode is blown it could represent a low.

mon methods. It is a portion of a ROM chip. There are three addresses and at each location there is a byte of data burnt in. The addresses are the rows and the bytes are the columns.

The layout is similar to the way a static RAM is made. All of the bits in the addressed byte are on one chip. In the C128, the system ROMs are either laid out as 16K × 8 or 32K × 8. Figure 7-1 shows three addresses with their respective bytes. The addresses, of course, are connected to the ad-

dress bus while the bits are attached to the lines of the data bus, D7-D0.

Each row has its own special address. Each column is connected to every row. Between the row and a column, a diode could be wired in. There is then one microscopic diode between each row and every column, making each diode represent a bit. In Fig. 7-2, row address 0010 has eight diodes attached. The cathodes of the diodes are connected to the row side. The anodes of the diodes are wired

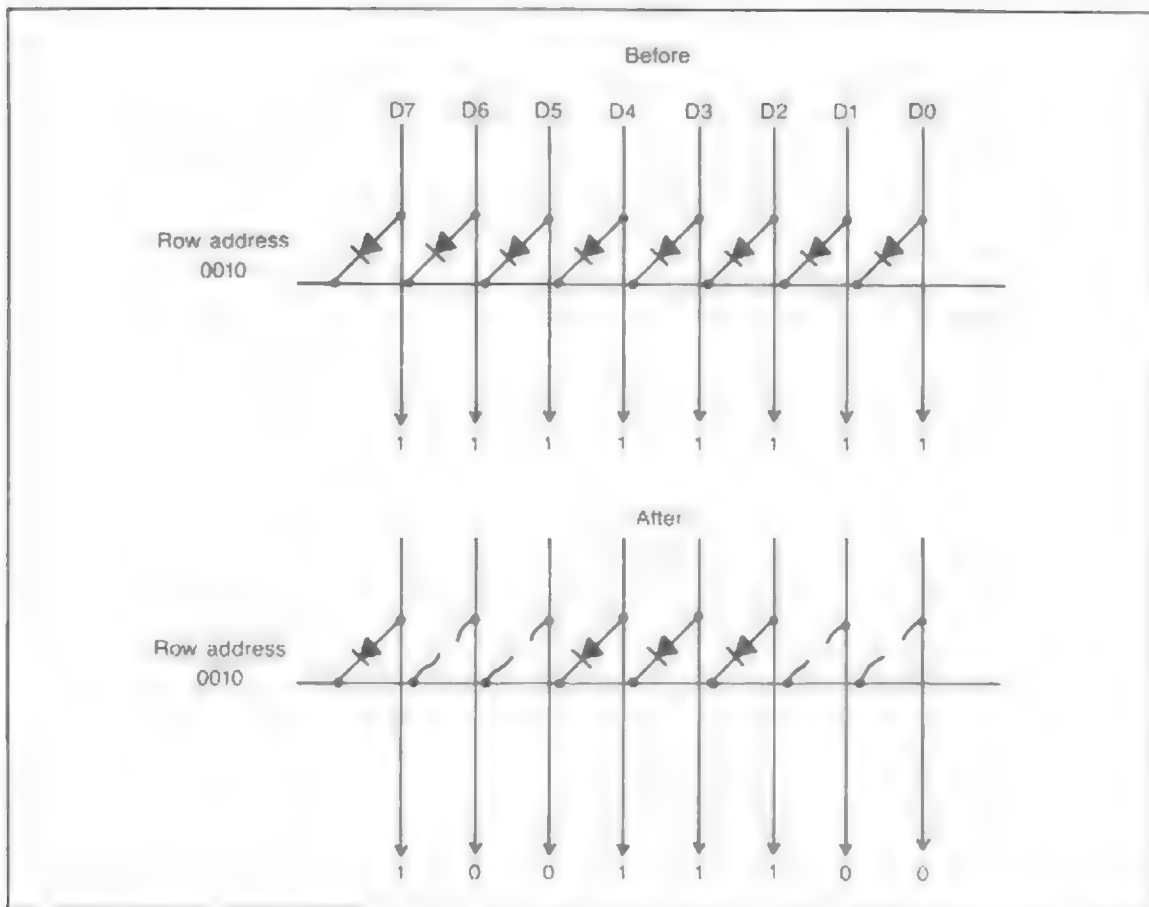


Fig. 7-2. A row location could start out life with all eight diodes intact. Its contents would then be 1111 1111. If some diodes are then blown the contents would change to 1001 1100.

to the column side. While all eight cathodes are connected in common to row 0010, the anodes are attached separately to the eight columns.

The columns in turn are attached to the data bus lines. When address 0010 is dialed up, the eight diodes will conduct, cathode to anode. According to the wiring, the conduction of the diodes could be code for a binary signal 1. The columns will then output 11111111 since all of the diodes are intact and conducting when addressed. The 1's will then flash over the data bus to the MPU.

Suppose you are the one coding the ROM. You do not want address 0010 to output 11111111. You want address 0010 to output binary 10011100. In order to effect the change, you must alter some of

the diodes to produce the change in signal. An analysis of the change shows that you want to alter bits D6, D5, D1 and D0. Bits D7, D4, D3 and D2 should be left alone because they are already outputting 1's. But you want 0's installed in bits D6, D5, D1 and D0.

The technique is to apply a large voltage onto the diodes where the 0's are to be installed. With the application of the destructive voltage, the diodes to hold 0's are blown. They become open circuits. They can no longer conduct like the diodes that code the 1's. With the conducting diodes representing 1's and the blown diodes representing 0's, the data at address 0010 becomes the desired 10011100.

Although this procedure is fine for the experimenter on a workbench, during actual ROM

manufacturing nothing so slow can be used. After the program to be etched onto a ROM is written, a large printed circuit pattern is made. The junctions that are going to contain a diode or transistor are clearly defined on the pattern. They will be the row-column locations that will be outputting 1's. The row-column locations that are to designate 0's are left open. Once the pattern is made, it is then reduced photographically and used to manufacture large quantities of the ROM.

In the C128, U's 33, 34 and 35 started life as identical chips. Then the operating systems were burnt into the chips. They each assumed new identities. They are still addressed and have their data output in the same way, but the data they output is different.

BLOCK DIAGRAM

There are four main parts to a typical ROM, besides the power in and ground. For example, look over the U18 the character ROM. In the ROM is code to produce on the screen two complete character sets. One set is a group of 256 uppercase and graphic characters. The second set is another group of 256 upper/lowercase and graphic characters. There will be more detail on the ROM contents in the video Chapters: 20 and 21. U18 is shown in Fig. 7-3—the third Test Point Chart. There are 24 pins on the chip. The pins are all connected to microscopic circuits that are completely inaccessible to you. Even though the circuits are too tiny to work with directly, you can make tests accurately from the pins on the chip.

In the block diagram in Fig. 7-4, note the memory matrix. The memory is organized as 4096×8 . This means 4096 byte-sized locations are in the ROM. The locations are figuratively stacked in a tall skinny pile with location 0 at the very top, and location 4095 at the bottom. Actually, the matrix is laid out in a grid fashion, like dynamic RAM, but it is wired as a tall pile, like the static RAM, so we should consider the wiring, not the physical way it is produced.

In order to address 4096(4K) individual locations: 12 address pins must be on the chip. 4096 different combinations of 12 bits are possible. To

choose one address out of the 4K locations, 12 address bits must be applied. For instance, to address location 185 you'd send bits 000010111001 out over 12 address lines. Those highs and lows will activate location 185 and leave the rest of the locations shut down.

The 12 address lines A11-A0 are the connections made to pins 18, 19, 22, 23, 1, 2, 3, 4, 5, 6, 7 and 8. The lines enter the package and are connected to an internal circuit called the address decoder. In the decoder, the highs and lows are evaluated and control voltages generated. These voltages are processed into the memory matrix and activate the single location that was desired.

Once the location is accessed, the voltage applied at the row address follows the wire pathways as seen in Figs. 7-1 and 7-2. The paths are through components, like diodes, that are connected from the row line to the column lines. The pathways with intact components pass the highs. The pathways that were blown, or are missing diodes, do not pass a high—they maintain a low. The byte of highs and lows travels out the columns to the next stage. These bit holders can only output highs and lows—they are "read." They cannot receive highs and lows, or be written to. The bit patterns are permanently burnt into the memory matrix.

The next stage in the ROM is a set of eight three-state buffers. Buffers are covered in Chapter 10, but for now consider a buffer to be an amplifier that can be turned off and on with the aid of its three-state characteristic. The buffers are turned off and on by another circuit in the ROM which will be discussed next. When the buffers are on, they pass the highs and lows from the eight matrix columns to the eight pin connections that attach to the data bus, D7-D0. These are at pins 17, 16, 15, 14, 13, 11, 10 and 9.

Between the incoming 12 address lines and the eight data lines, 20 of U18's pins are accounted for. Pins 24 and 12 are +5 volts and ground. This leaves two pins unaccounted for. They are pins 21 and 20. Pin 20 is called *CS1. The signal *CHAROM (the character ROM select signal) from pin 46 of the PLA, is applied to pin 20. It so happens that U18 shares addresses with some I/O functions. When

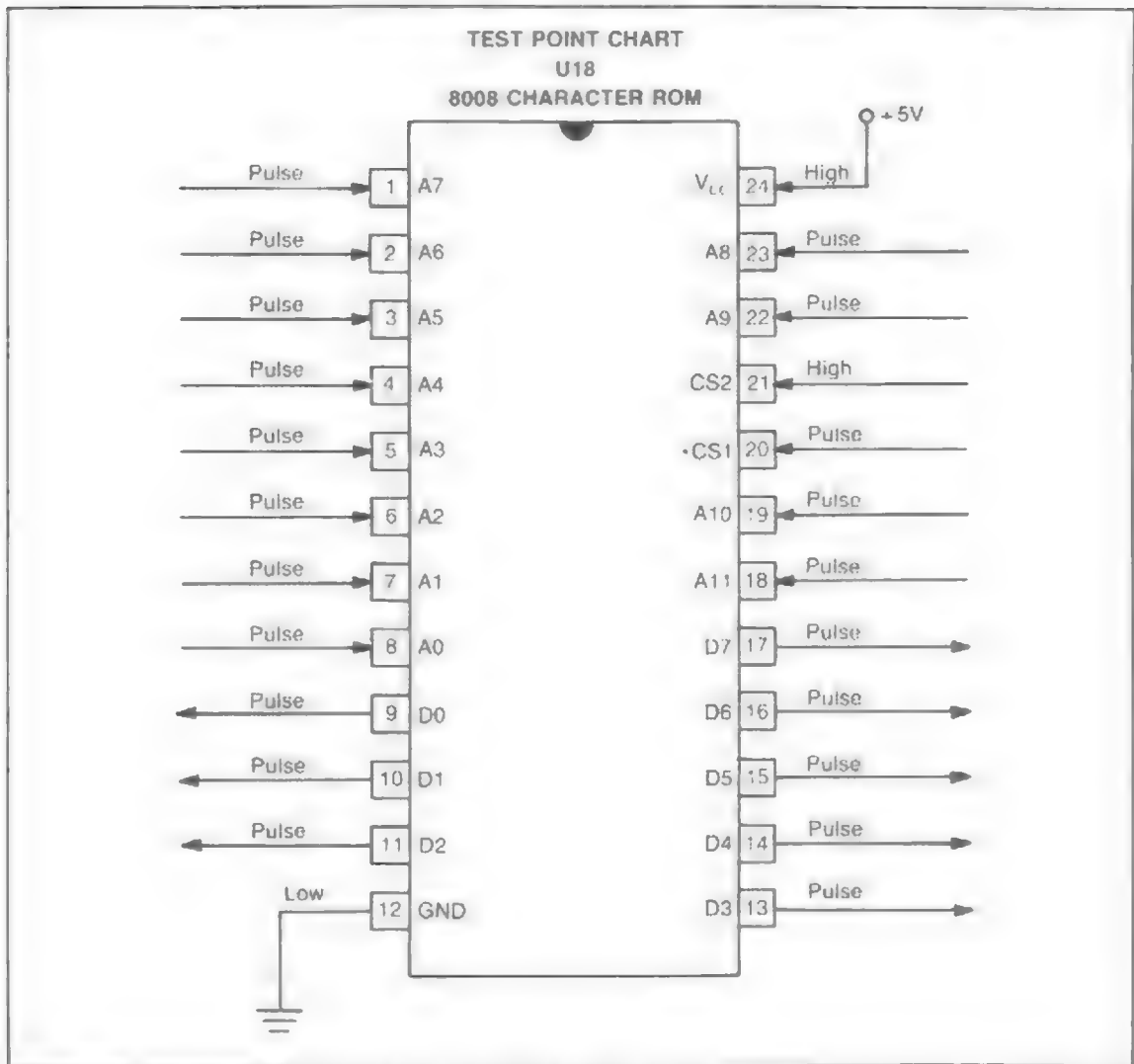


Fig. 7-3. U18 is the character ROM Test Point Chart. The logic probe should show all pulses except for the chip select, CS2, the +5 volts and ground.

U18 is selected the I/O is switched off. When U18 is not needed, the I/O uses the addresses. Pin 20 does the selecting.

Pin 21 performs another job. Inside this ROM chip are two 4K sets of characters, totaling 8K. Only one 4K set, however, is available at a time. Why two character sets? One set of 4K is used in C64 mode, and the other 4K set is used in C128 mode. Pin 21 is connected to the 128/64 signal coming from

pin 47 of the MMU and pin 15 of the PLA. When you are in C64 mode, then the signal automatically chooses the C64 character set. If you are in C128 mode, then the C128 character set is chosen.

Character ROM Contents

The character ROM, in one 4K set, has 4096 bytes of permanent memory. Internally the set is

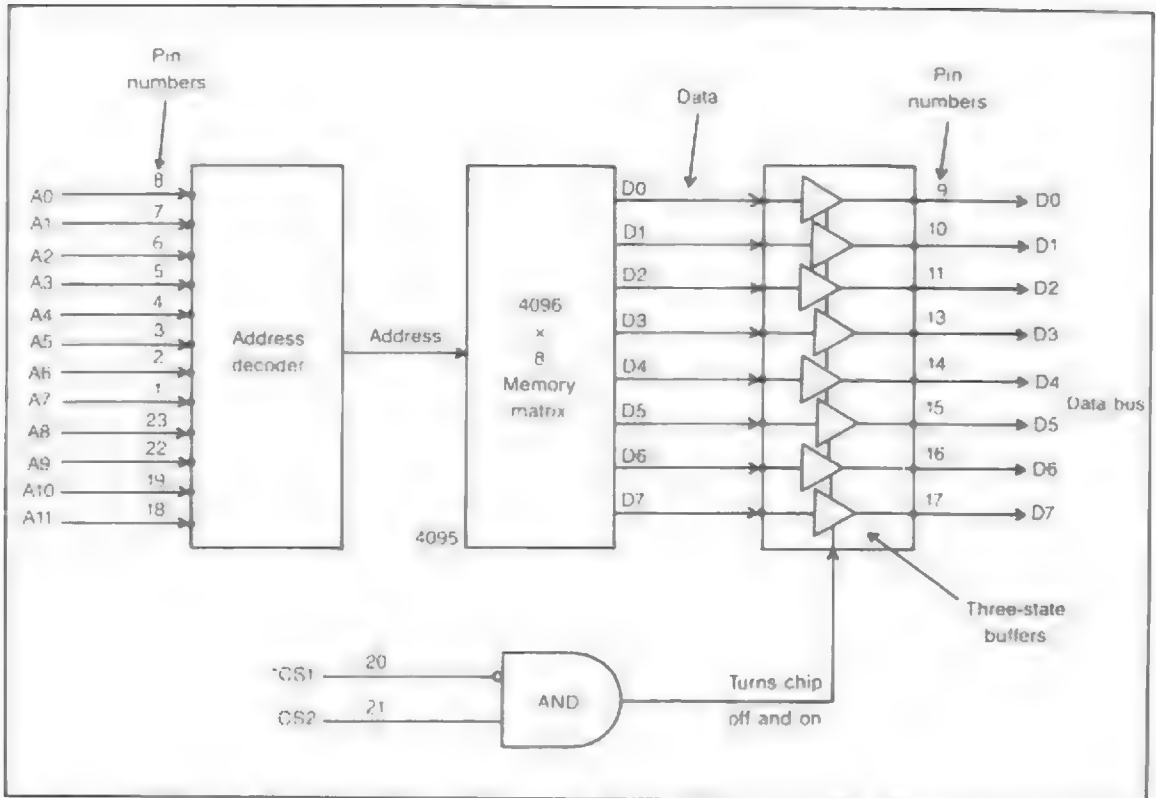


Fig. 7-4. The inside of the ROM is fairly straightforward. The chip-selects turn on the chip, the address lines locate the desired register, the register outputs to some buffers and the data heads out over the data bus

addressed from 0 to 4095. In the C64 mode the ROM is addressed from 53248 to 57343. In the 4096 bytes of memory there are 512 characters. Each character needs eight bytes to be stored ($512 \times 8 = 4096$).

Each displayed character is composed of a grid of dots in a character block that is eight light dots wide and eight dots high (see Fig. 7-5). The 64 dots can be turned off and on to produce a character pattern. A high from a bit in the ROM turns on a light dot and a low turns off the dot.

The output of a character ROM leaves the chip through the data bus, D7-D0. The output goes to VIC. The VIC receives a character shape in that way and processes it into the TV display.

The first character in ROM is at address 53248 and extends eight bytes to 53255. It is the character, @. It is an uppercase character. The 512 charac-

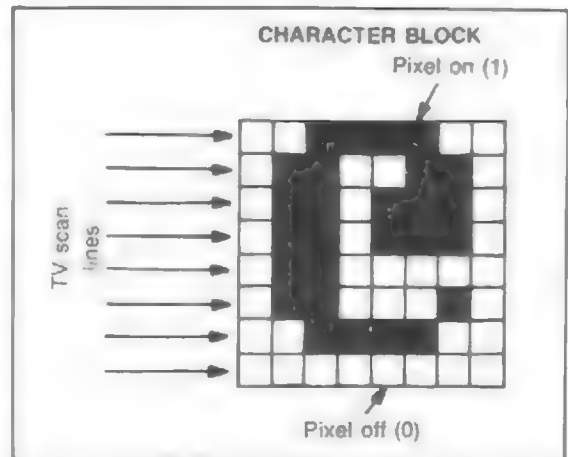


Fig. 7-5. A character in the display appears when a grid of eight light dots across by eight dots down has some of its dots lit and others turned off. Here is how the character @ is formed.

ters are contained with the following decimal addresses for each category of characters:

Uppercase/Graphics Set

- 53248 Uppercase characters
- 53760 Graphics characters
- 54272 Reversed uppercase characters
- 54784 Reversed graphic characters

Uppercase/Lowercase Set

- 55296 Lowercase characters
- 55808 Uppercase and graphic characters
- 56320 Reversed lowercase characters
- 56832 Reversed uppercase and graphics characters

Checking Out the Character ROM

ROMs are very sturdy in comparison to other chips, but trouble can strike them. If some bits on a ROM should get destroyed, then incomplete characters will appear on the screen. It is easy to check out the characters on the ROM if character trouble should happen.

In C128 BASIC, a little three line program can be written. The character ROM can be accessed through Bank 14. (There will be more about banking in Chapters 14, 15 and 16.) The program will PEEK the contents of any or all of the ROM bytes. You just have to specify which bytes you want to know the contents of. The binary contents will be returned in decimal. You can then obtain the patterns of 1's and 0's from the Byte Sized Conversion Table in the back of this book.

For example, suppose that you want to see the bits making up the @ character. The first thing to do is to enter the PEEK program for the eight bytes in which the character is stored in the ROM. It is stored at addresses 53248 through 53255. The first line accesses Bank 14 and the last line moves the program back to Bank 15. The following is a simple program for the C128 mode:

```
100 BANK 14
200 FOR C = 53248 TO 53255: ?
    PEEK(C);:NEXT
300 BANK 15
```

When the program is run it will return **60 102 110 110 96 98 600**. The bit patterns will look like the following in an 8 × 8 matrix that one character block on the screen displays:

Decimal	Binary							
60	0	0	1	1	1	1	0	0
102	0	1	1	0	0	1	1	0
110	0	1	1	0	1	1	1	0
110	0	1	1	0	1	1	1	0
96	0	1	1	0	0	0	0	0
98	0	1	1	0	0	0	1	0
60	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

If you look closely at the bit pattern you can see the @ defined. Figure 7-5 shows what the screen looks like with the 1's lit and the 0's unlit.

You can also access the character ROM in machine language with the Monitor, or in C64 mode in BASIC if need be—though it is more of a programming job. Without the Bank command you must write lines to switch in the character ROM and switch out the I/O hardware that is sharing the addresses. Then you must transfer the ROM bytes to RAM. Then the ROM has to be switched out and the I/O switched back into the memory map and BASIC. Then the ROM bytes can be printed on the screen in decimal.

Besides this test method, you can check out the ROM with a diagnostic program like the 64 Doctor mentioned earlier in the book.

THE REST OF THE ROMS

The character ROM is a special type that supplies display materials. The other four ROMs: U32, 33, 34 and 35, deal mostly with: the Kernel, the Monitor and BASIC. The empty ROM socket, called U36, is there for expansion. The socket is wired; complete with 32K worth of addressing, eight data bus lines and power. Any ROM plugged in will start operating immediately and can be contacted directly

from the keyboard. It can be thought of as a special internal port.

Chips are produced for the socket that enhance the operation of the C128. One chip is advertised by Utilities Unlimited Inc., 12305 N.E. 152nd Street, Brush Prairie, Washington 98606. They call it The 128 Superchip. It gives 32K of utilities, such as File Copier, Nibbler, Track & Sector Editor, Screen Dump and 300/1200 baud Terminal Program. There will be more of this type of chip available as time goes by.

The ROMs already in the C128 are operating systems (OS). They are all 28-pin DIPs. They contain all the programs that run the computer. Whenever the MPU makes a move, it checks with the ROMs for directions on how to do it.

Internally the ROM structure is quite like the character ROM. They are powered with +5 volts at pin 28 and returned to ground at pin 14, as shown in the Test Point Chart of Fig. 7-6. They all connect to the data bus D7-D0 at pins 19, 18, 17, 16, 15, 13, 12 and 11. Though the character ROM has only 12 address pins, these four other ROMs have either 14 or 15 address pins. With 14 pins A13-A0, 16K of addresses can be contacted. With 15 address pins A14-A0, the ROM has 32K of programs.

In different model runs of the C128, U32 and U34 have been either 16K or 32K ROM. U33 and U35 are usually 16K. As time goes by, however, be prepared for either 16K or 32K in any of the ROM sockets.

The 16K ROM has the power, addressing and data pins, and a few other pins of note. Pin 1 is a Programming Voltage connection but is unused in the C128. It is connected to +5 volts and forgotten. Pin 20 is \bullet CE, a chip enable. It is active when a low is applied. It is tied to ground, which is a low. The pin is thus always active. Pin 22 is \bullet OE, an output enable. It is active low. It receives a signal from the PLA. On U32, pin 22 receives the signal \bullet ROM1. If this 16K chip is in any other chip position, then it will receive the appropriate signal. For instance, if U35 is a 16K it will receive the signal \bullet ROM4 from the PLA.

The 32K ROM also has other connections. Pin 1 is a Programming Voltage pin that is tied to +5 volts and forgotten about as in the 16K ROM. Pin 20 is also a chip enable, tied to ground and forgotten about. Pin 22 is the output enable and receives an appropriate \bullet ROM signal from the PLA to select the chip. What actually are these signal inputs doing?

In Fig. 7-7, a timing diagram is shown for both of these ROMs: the 16K and the 32K. On the top the addressing bits are shown. They enter all of the address pins. The bits select one of the byte-sized locations on the ROM.

At the same time, into pin 20 of the chips enters the chip enable signal. It is actually always low since the pins are all tied to ground. The pins are active low. Also at the same time the output enable signal is entering at pin 22. It is coming from the PLA. It starts off high and then, as the address bits and the chip enable signal are working, it falls low. This activates the chip. Since the ROM can only output data, it does so to the data bus pins.

ROM Contents

In Chapters 14 and 15, the ways in which the PLA and MMU control the RAM and ROM is covered. They turn the ROMs off and on to get the best possible use of the group. U32, also called ROM1, is the container for the C64 mode operating system. It can be 16K or 32K according to what model run C128 you are using. In either case, 16K of ROM1 contains the BASIC and Kernel programs—split up about 8K each.

On the BASIC ROM are hundreds of small machine language programs that respond to all the BASIC statements and all the requests for calculations and data manipulations. In C64 mode, the ROM is activated by approximately 65 special keywords. The keyboard upper and lowercase characters, plus the digits 0-9, are used to produce these 65 keywords. Some of the other symbols are also used to produce the BASIC keywords. The keywords are the C64 commands you are familiar with: for instance, CHR\$, DATA, GOTO, etc. When you enter these keywords, the BASIC ROM is addressed and runs a lit-

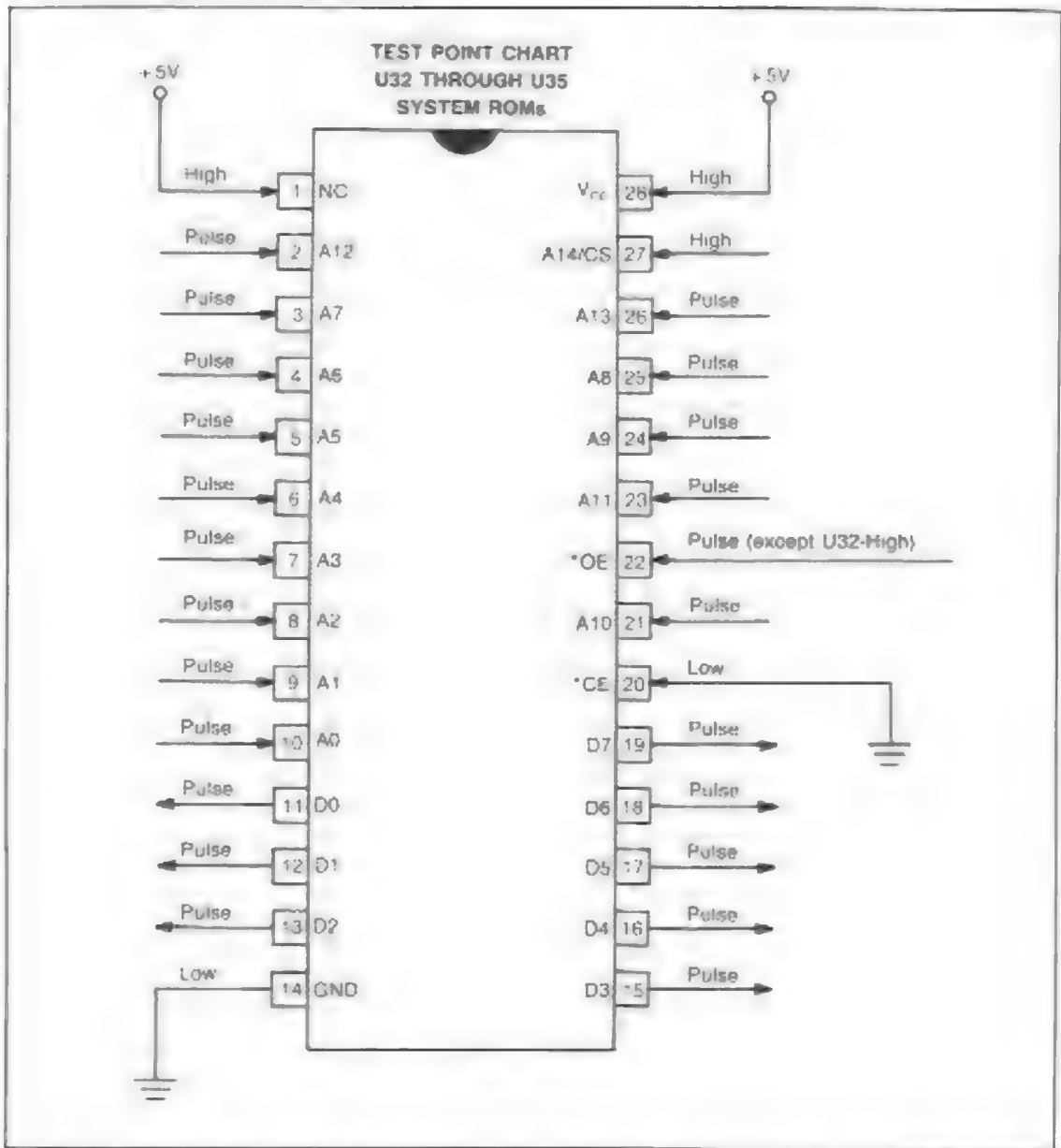


Fig. 7-6 All four of the system ROMs will show the same results on the logic probe except for U32's pin 22 which will show a high instead of a pulse.

the program that performs the chore that the keyword is designed to do. The keywords and other programming details are found in the System Guide that came with the C128.

Kernel ROM

The BASIC ROM is a collection of a few hundred little programs that execute the keyword commands of the BASIC language. The Kernel contains

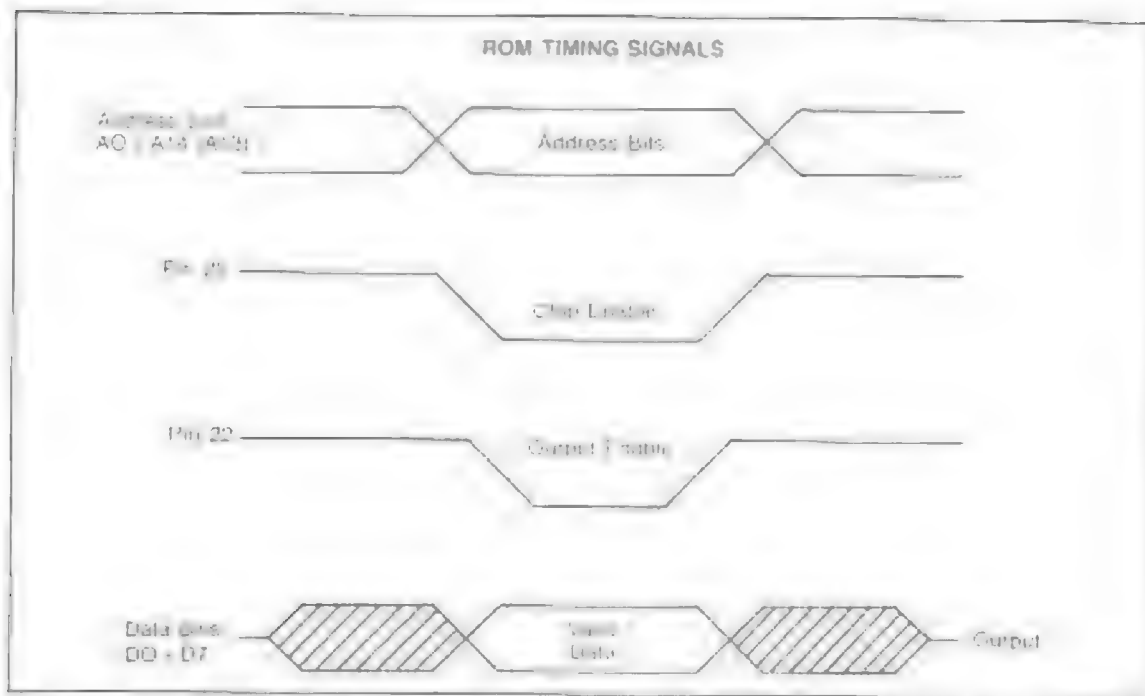


Fig. 7-7. The timing signals for the ROMs are all very much the same. As the ROMs are addressed, they receive lows at the chip enable and output enable pins 20 and 22. This causes the addressed register contents to be output onto the data bus.

the programs that perform the rest of the operating system's duties. The Kernel, first of all, sets up the machine when you first turn it on. It initializes all the registers on all pertinent chips.

Secondly, the Kernel activates the screen editor for the C64 activity. The screen editor takes the signals generated from striking the keys and has them installed in their respective character blocks on the TV screen. The screen editor also controls the cursor. It also responds to commands like delete and insert.

Thirdly, the Kernel provides access to the operating system's group of programs in the ROM. It does so by means of special Commodore jump tables. These jump tables are very important to users of the VIC 20, the C64, and now the C128. As will be explained in the next section, the jump table gives these computers a considerable amount of longevity. As time goes by, the computers will not tend to become obsolete as quickly as new operating systems are introduced. The jump tables allow your older ma-

chine to adapt to a lot of the new subroutines in a new operating system.

When you first switch on the C128 in C64 mode, the Kernel goes into its initialization sequence. It sets a section of RAM called the *stack*. Then it clears the decimal mode of operation. Next the Kernel checks the ROM cartridge holder circuits. If a ROM cartridge is plugged in, then the control of the machine is switched to the cartridge system. If the holder is empty, then the Kernel retains control and continues its initialization procedures.

The Kernel then turns its attention to all the I/O devices and their registers. They will be discussed in Chapters 19 to 23. The serial bus is initialized. CIA1 is set to constantly scan the keyboard. The 60 Hz timer in each CIA is turned on. The SID chip is cleared and the BASIC memory map for the C64 mode is selected. The cassette motor is turned off.

As a servicing checkout, the Kernel contains a special diagnostic program that exercises the RAM that the C64 mode will be using. The Kernel runs

the diagnostic at the beginning to make sure that the RAM chips are intact. Once the RAM "checks good," the Kernel then sets the memory pointers for the top and bottom of RAM. Then the first page in RAM, page 0, is initialized. The tape buffer is then set.

The Kernel then stores some special addresses in RAM at specific locations. These addresses are needed during some I/O activities. Another little jump table is installed by the Kernel in the low section of RAM. The TV screen is cleared. The screen editor has its variables reset.

All of these steps take care of the original house-keeping duties that the Kernel must perform before you begin your BASIC programming. Once all this is accomplished, BASIC is addressed and you are READY. The Screen Editor program in the Kernel will now work with you to display your keyboard input.

Jump Tables

As you can see, the operating system in the C64 mode is detailed and complicated. It is even more so in the C128 mode. The BASIC and Kernel C64 mode programs in U32 are also critical. If one vital bit in an important byte becomes defective, then the entire 16K of C64 ROM could be rendered useless.

Like the BASIC section, the Kernel part is also filled with many operating programs. The Kernel's programs, though, are not activated by BASIC keywords. The Kernel's programs are turned on by machine language code numbers. Another point to remember is: though the BASIC programs can be run with keywords, the BASIC output is in machine language, like the Kernel output is. The machine language output of BASIC and Kernel are instructions and data that the rest of the computer can understand and process.

The Kernel is divided into two main sections. The control programs are located at the beginning addresses of the ROM chip. Consisting of a few dozen input/output routines and a number of other management programs, these routines take up most of the addresses on the chip. The jump table code is located near the end of the address list. The jump

table is a group of locations with each location full of data as seen in Fig. 7-8. The data in each location consists of an address. Each jump table address is a starting point of one of the programs stored in the beginning of the Kernel ROM chip. The chip works like the following. When you write a machine language program, and you want to call a particular Kernel routine, then you address the location in the jump table that holds the starting address of the routine you want to use. As a location in the jump table is addressed, it is activated. Its contents—the starting address of the desired routine—is output. That starting address is, in turn, put on the address bus and the routine is thus contacted.

The question arises, why bother with this indirect way to contact the routine? Why not just address the Kernel locations directly? The answer is that the Commodore designers want the present models to be compatible with, and operate with, future models.

As time goes by, improvements in the operation of the C128 will continue. Even a small change could make a newer operating system useless in an older model. In the interest of having new OSs work in older machines, the jump table scheme is utilized. The jump table is designed to standardize Commodore models with respect to input, output, and memory management in the following way.

During expected changes, the actual routines are not expected to change much. But even if they do, they should still run the 8502 MPU and its DRAMs without undue complication. What is expected to change are the addresses of the routines. If the addresses of the routines change, literally thousands of programs on tape or disk will become obsolete. A lot of your programming won't work with these changed OSs. That is where the jump table comes in.

When a new Kernel chip is manufactured, the jump table will be located at the same chip places at the back addresses as its predecessors. The contents of the locations, which are the addresses of the routines, can be changed. That way your old programs will still address the same old places in the jump table. The new address contents will then point

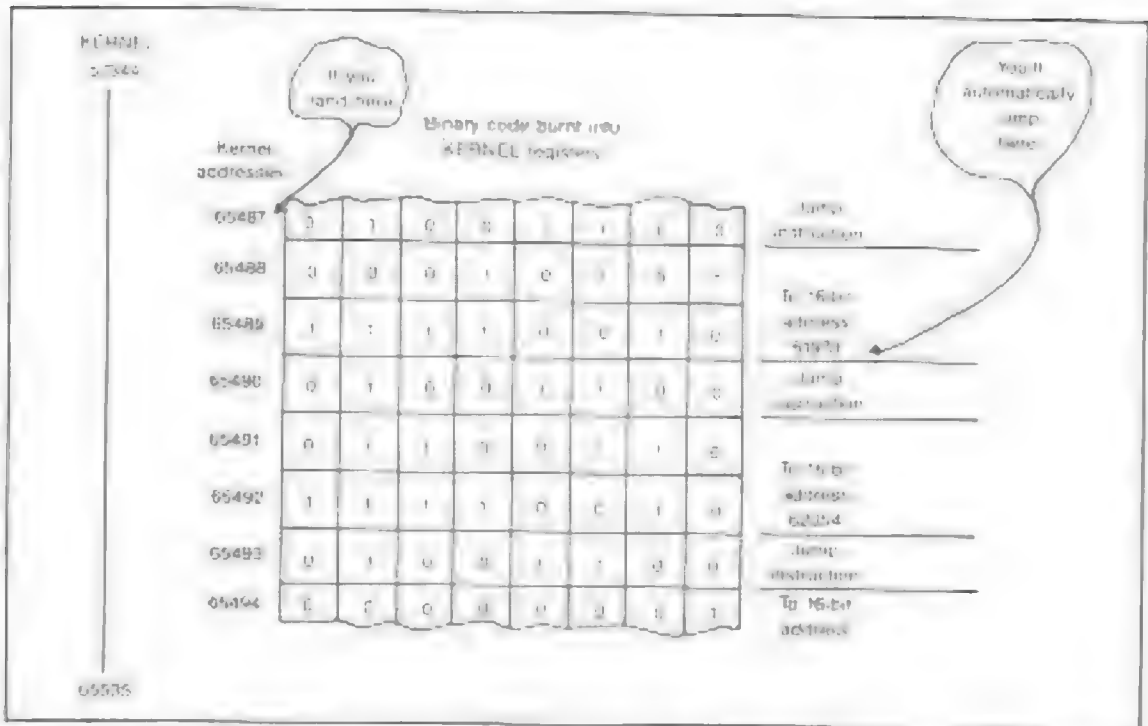


Fig. 7-8. At the highest addresses of the Kernel ROM is a special jump table. The table consists of a number of three-byte data groups. Each three bytes is made up of a one-byte jump instruction and a two-byte address. If you address one of the data groups, you are instructed to jump to the stored address.

to the new address of the routine. You won't even know the difference as the program is run.

I did not mention that the jump table holds some program addresses not found in the Kernel. These off-chip addresses are located on the BASIC ROM. That is because the BASIC ROM chip contains a lot of programs that are very useful. Instead of writing some of these routines, the programmer saves time by just putting a starting address in his program that goes to the jump table. The jump table, in turn, addresses a program on the BASIC ROM. The program is then called and run saving programming time and effort.

Diagnostic Programs

When the C128 is completely down, then it can't help you locate the trouble. When it is unconscious, then you must use the electronic test techniques re-

quired. This includes using the Test Point Charts, the Master Schematic, etc., with test equipment such as the vom and logic probe. However, if the C128 is not unconscious, but is operating somewhat erratically, then you can use programming tools like PEEK and POKE to check it out.

A short test routine using PEEK was discussed in the Character ROM section. With it, every byte in the Character ROM can be examined. With an eight byte examination, a character can be derived. The bits can be drawn on paper to depict the character. If a few bits have been destroyed, then those bits can be pinpointed. The chip is then replaced.

All of the ROMs can be read with PEEKs in a similar way. In fact, most of the memory map can be tested by PEEKing the map locations. The ROMs can only be read with PEEKs, but most of the rest of the map can not only be PEEKed but also written to with POKEs.

The C128 has 16 default memory map layouts. In BASIC they are called BANKs and are numbered from 0 to 15 in decimal. To set up any one of the 16 layouts, all that is needed is to write a program line with a BANK number. In the character ROM PEEK, the first line was BANK 14. BANK 14 is the only layout that includes the character ROM. Table 5-1 lists the 16 BANK arrangements and the type of memory that is included in each BANK.

Note that in the character ROM PEEK routine that the banking can be changed from program line to program line. That is how an MPU like the eight-bit 8502, with only a 64K addressing capacity, can actually address 128K in the same program with the aid of banking.

Anyway, when the C128 is conscious, the best place to start testing is in the ROMs. Reading the ROMs is the first test of choice. The ROMs are the chips in control. If you can get the MPU to read from the ROMs, you know they are operating. When you first turn the computer on, the MPU is built to automatically read the start bytes of the ROM operating program. The computer signs in by printing its READY message on the TV screen. If this happens then the MPU is successfully reading from the ROM system. Admittedly, the MPU is only reading a dozen or so bytes, but reading is going on. The ROM can be read, and if any discrepancies are present, then a PEEK routine picks it out.

Therefore, when the ROM signs in, you can use diagnostic programs for testing the residents of the memory map. The computer is conscious. There are a lot of different ways the ROM can be tested for intact routines. The easiest way to go is to purchase diagnostic software like the 64 Doctor that has been selling for a number of years. Perhaps you would like to write a ROM diagnostic, somewhat like the one in the 64 Doctor, for your C128.

The idea is to test every bit in the ROMs. In general a program must do the following. The test is to read each byte in a ROM chip, byte after byte. The way it has been done is to run each bit in each byte through a 16-bit shift register. The shift register is to XOR each incoming bit from ROM with its 6, 8, 11 and 15 bits. As each bit passes through, the value of the shift register changes.

The test number in the shift register is called a CRC for *Cyclic Redundancy Check*. Because the register is 16 bits, and four hex numbers code 16 bits, the test number is called a four-digit CRC. This is an error checking test used by programmers to check the transmission of programs. Troubleshooters use it to read the ROM.

At the end of the ROM read, the shift register is left with a hex value. This value, which is predetermined by running the test on a proven C128, is then printed on the screen. If the number on the screen matches the number of the known-good C128, then the ROM has been read and is okay. Should a chip not finish up with the prescribed number, then the chip is probably a damaged one.

A ROM can be disabled if any shorts or opens occur in the burnt-in program bytes or control circuits. A test program as described should pick out the bad ROM quickly.

A Short ROM Routine

In the same way that the character ROM was read, you can also read almost any section of ROM with a short BASIC program. For example, in the higher addresses of ROM3 are some jump table numbers. The contents of the table bytes contain an address. At decimal 44917, the first byte of a three-byte routine called JMP GPLOT is located. If you desire to access the three-byte routine, and read the contents of all three bytes, it can be done with the following short BASIC program:

```
100 BANK 14
200 FOR X=44917 TO 44919
300 ?PEEK(X);
400 NEXT
500 BANK 15
```

When you run the program, the C128 mode screen will show the contents in decimal:

```
RUN
76    251    155
```

In the same way you can access almost any ROM location or group of locations. This is a valuable test

technique. In Chapter 16, the C128's memory maps are covered. With the aid of that information, you can run a test set of readings while your C128 is operating okay. These readings, which will be the contents of the ROM locations, can be printed out

on a sheet of paper. That way, if you ever suspect that a ROM has become defective, then you could reprint the ROM and compare the latest readings with the proven readings. If they do not match, then the ROM under test has become defective.

8. Other Integrated Circuits

There are 63 integrated circuit chips in the C128. We have looked over the nine large chips, the 16 DRAMs, the color RAM, the character ROM and the four OS ROMs. That is a total of 31 chips, leaving a balance of 32 smaller support chips. The nine large chips, the character ROM and the color RAM will all have more detailed coverage later in the book. This chapter will deal with the 32 smaller support chips.

The support chips do many important jobs. Let's determine: what they are; the jobs they do; how to test them if they become suspects during troubleshooting. See Fig. 8-1, 8-2. In this chapter is a set of Test Point Charts for these chips. The charts give the logic state of each pin on the chip while the C128 is idling at READY.

THE 7406 HEX INVERTERS

There are four 7406 chips: U29 just to the left of VIC; U30 above the empty ROM socket at the top left of the board; U37 at the bottom of the board

to the right of the PLA; U63 above CIA2 at the left side of the board.

The 7406's are 14-pin DIP's. The word hex means "six," and there are six little inverters built into the chip. What is an inverter?

As you'll learn in Chapter 10, an inverter is sometimes called a NOT gate. If you inject a logic high into an inverter, the circuit changes the high to a low, and a low will emerge from the gate. Should you input a low into the inverter, it is changed to a high. Refer to Fig. 8-3.

Besides inverting the logic state, an inverter can act as a current amplifier and buffer to the incoming signal. It can raise the current level of the logic state so that the state will match to the subsequent circuits.

The 7406 is a TTL with six of these inverter stages. Figure 8-4 shows that each inverter stage uses two pins. Only one input and one output are connected to each NOT gate. The six inverters require 12 of the chip's 14 pins. The other two pins are used for a +5 volts power and ground.

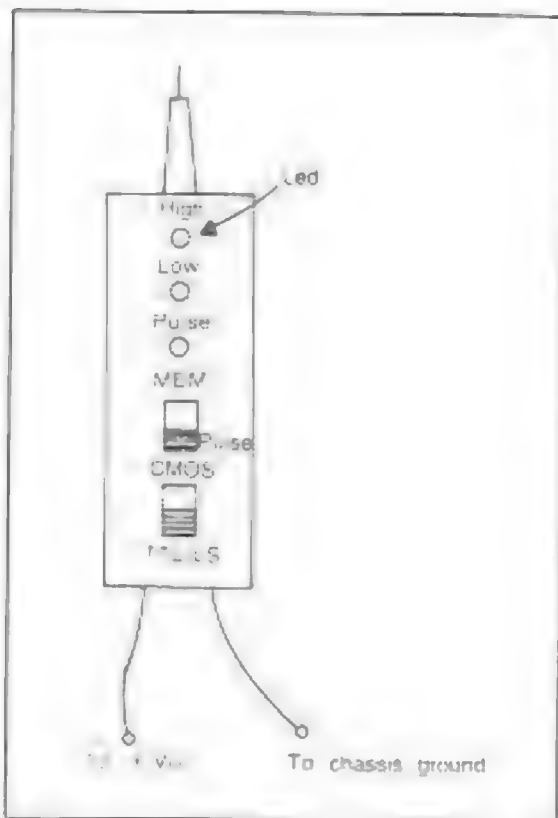


Fig. 8-1. The Test Point Charts show the logic states that should be present on all the chip pins. The logic probe is the device that has LEDs which reveal the highs, lows and pulses clearly. No light is on when a pin is tri-stateing.

The six inverters are all on the same chip, and all use a common power input, but they are otherwise independent of each other. The six gates on the chip are all installed in different circuits throughout the computer. For example, in the output lines of CIA2 there are two inverters from U63 and three inverters from U30. In the cassette motor line is one U30 inverter. There are four more U63, one U29, two U30 and a U37 in the 556 chip clock circuit. The clock is covered in detail in Chapter 17.

The NOT gates are sprinkled throughout the C128. The NOT has a schematic symbol. It is a triangle with a circle at the output point. If you look over the Master Schematic, you will see the little NOT gate symbols. As you look at the NOT gate you must realize that whatever signal is input at the

flat side will be inverted and amplified as it emerges from the pointy end.

With the aid of Figs. 8-1, 8-2, 8-4, Table 8-1, and a vom or logic probe, you can check out the hex inverter quickly. Each inverter has its input and output. The only type of signal that should be present at the inputs and outputs are logic states. The 7406 has no three-state capabilities, so there should not be any three-state conditions on any of the pins under ordinary circumstances. The input pins, shown by the arrows in Fig. 8-4 are 1, 3, 5, 9, 11, and 13. The respective outputs are 2, 4, 6, 8, 10 and 12. Pins 14 and 7 are power.

The first check with the vom should always be pins 14 and 7. Positive 5 volts should be on 14 and 0 volts or ground on 7. If either voltage is incorrect, then it is a clue. For example, suppose that there is no voltage at pin 14. If there are +5 volts on the surrounding chips but not on 14, then chances are that the copper trace on the printboard to pin 14 is broken or has a corroded connection. When the quick-check at 14 shows power is present, then the individual inverter stages should come under scrutiny.

Test the voltage at each input. A high is considered +2.0 volts or higher. If the input is a high, then the output should be a low. The low on this chip (it's a TTL) is considered +0.8 volts or lower. When the input is a low, then the output should have the opposite state, or a high. All six of the inverters on the chip should follow the same logical approach: input high, output low; input low, output high. If the input should be a pulse, then the 7406 should produce an inverted pulse output.

If there is a discrepancy from this pattern, you have a clue. The inverter with the wrong output could be shorted or open. A dead short in the stage would place the input voltage onto the output. An open circuit would place the surrounding voltage on the output. This could be any voltage, including a three-state condition.

The inverters are all individual and are connected in many circuits over the board. If you find any of the inverters not connected to any circuit, you will find voltages on the pins. Unused gates usually have their pins tied to +5 volts or ground

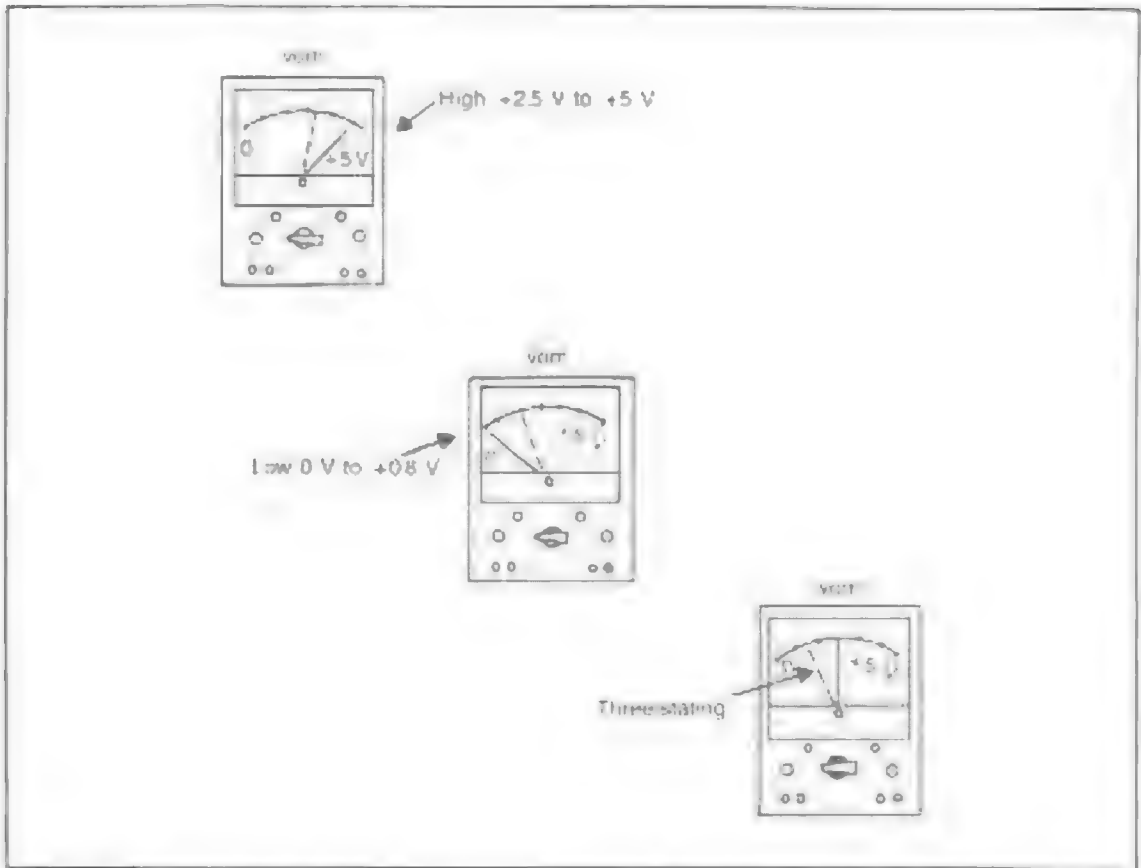


Fig. 8-2. An alternative to the logic probe is the voltmeter. It shows logic states by means of voltage readings on the pins. It reveals highs, lows and triesteating pins. The voltmeter's drawback is, it does not show pulses.

to keep them out of the activity, and this prevents them from causing circuit interference during normal processing.

THE 7407 HEX BUFFER/DRIVERS

U57 (to the left of VIC) and U29 (an inverter) and U60 (just to the right of the PLA) are three amplifier chips. A buffer/driver is simply an amplifier. The individual stages in each chip are drawn to look something like the inverter symbol, as in Fig. 8-5. A close look demonstrates the difference. No little NOT circles are on the pointy end of the triangle. Otherwise the symbols are the same. One input and one output connect each buffer. Table 8-2 provides

the logic states that should be present on U's 57 and 60.

On the Master Schematic, you'll find buffers in the video controller stages, four of them from U57 in the output lines of U24. If you look through the Master Schematic for the little triangles without NOT circles then you'll be able to find them easily.

Except for the circles that indicate an inverter, as far as troubleshooting goes, the 7407 is identical to the 7406. All the pin numbers and other characteristics can be considered the same. The only difference is that there is no signal inversion. The signal is still amplified so it will match into the circuits it is connected to, but the same logic state that enters a buffer stage will be output from that stage.

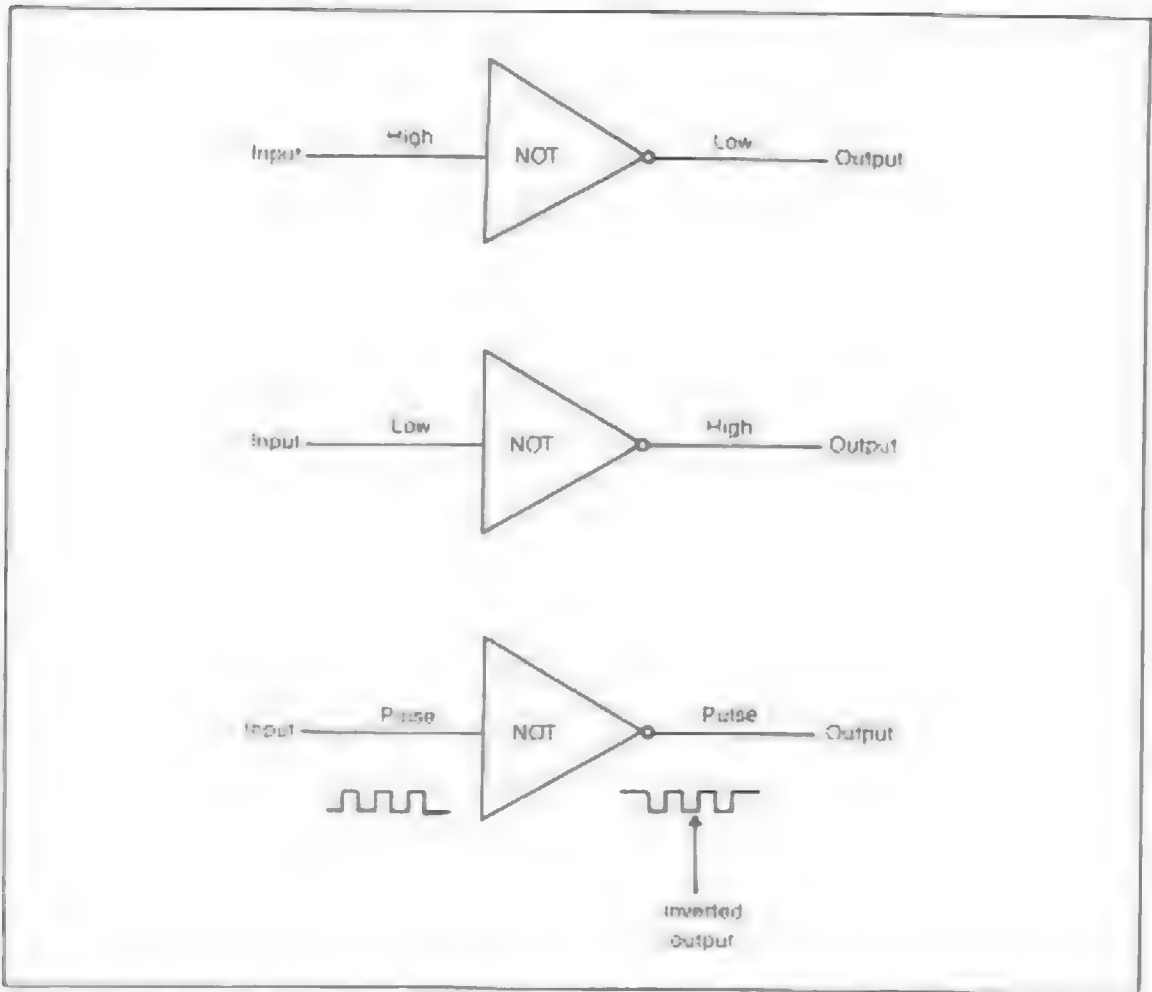


Fig. 8-3. Besides changing highs to lows and vice versa, the NOT gate will also invert a pulse that is high going to low.

THE 74LS08 QUAD 2-INPUT AND GATES

U8, a smallish chip at the bottom right in the center of five small chips, and U61, at the bottom right of SID, are the two 74LS08 chips. They are 7408 chips with LS extras. The L stands for low-power dissipation at the expense of switching speed. The S stands for Schottky. A Schottky diode is known for its high speed switching. By using both in a gate, the advantages work together.

The quad part of the name means four. Figure 8-6 and Table 8-3 show the four little AND gates on the chip. The 2-input means that each gate has two inputs in contrast to the inverters and buffers

that only need one input line. Even though there are two inputs into the AND gate, it only has one output. The two inputs are ANDed together to produce a logical output. More detail on AND gates is given in Chapter 10.

An AND gate requires a minimum of two inputs. It can have many more inputs but it still only uses one output. On the 74LS08 there are four gates. Twelve of the 14 pins are used to service the gates, three pins per gate. Pins 14 and 7 are used for +5 volts and ground like the 7406 and 7407.

The AND gate is built so that it will output a high if, and only if, both inputs are high, as shown

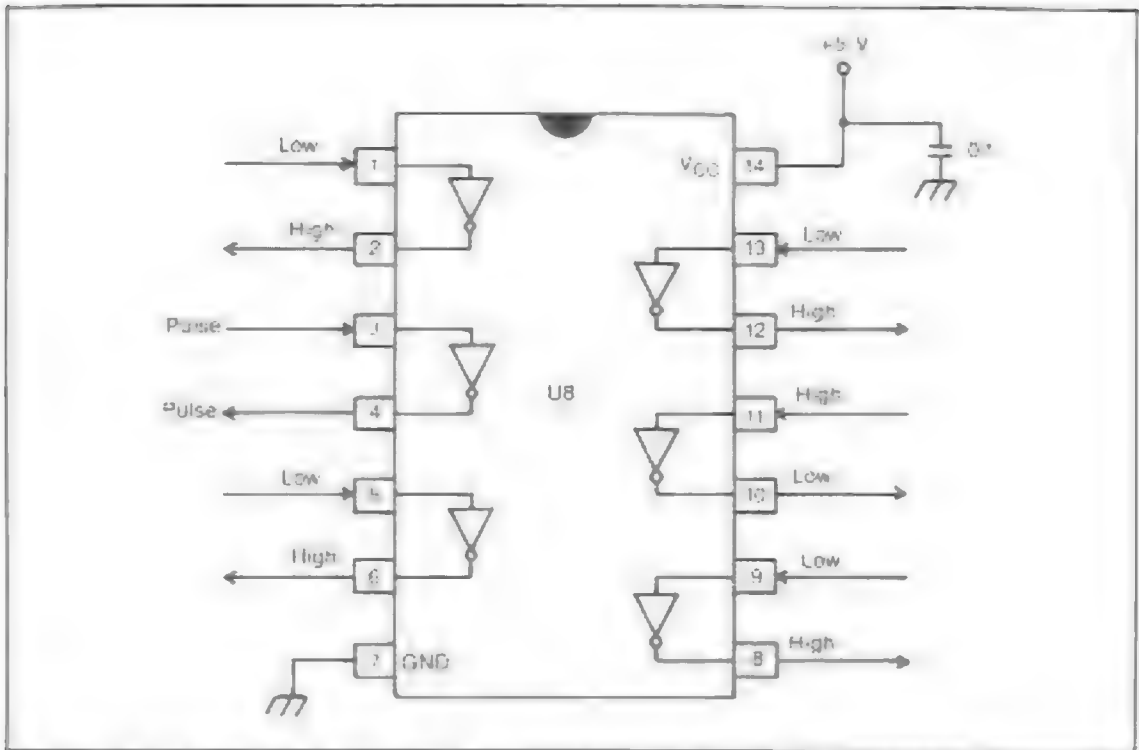


Fig 8-4 The 7406 is called a Hex Inverter. This means that it has six individual NOT gates on the chip.

Table 8-1. Four 7406 chips are in the C128.
Here are the logic states that should be present on each one.

Test Point Charts					
PINS	U29	U30	U37	U63	
1	Pulse	Low	Low	Low	
2	Pulse	High	High	High	
3	Low	Low	High	High	
4	High	High	Low	Low	
5	Pulse	Low	Pulse	Pulse	
6	Low	High	Low	Low	
7	Low	Low	Low	Low	
8	Pulse	0	Low	High	
9	Pulse	Low	High	Low	
10	Pulse	High	High	Low	
11	Pulse	Low	Low	High	
12	Pulse	Low	Low	Low	
13	Pulse	High	High	High	
14	High	High	High	High	

0 = No light

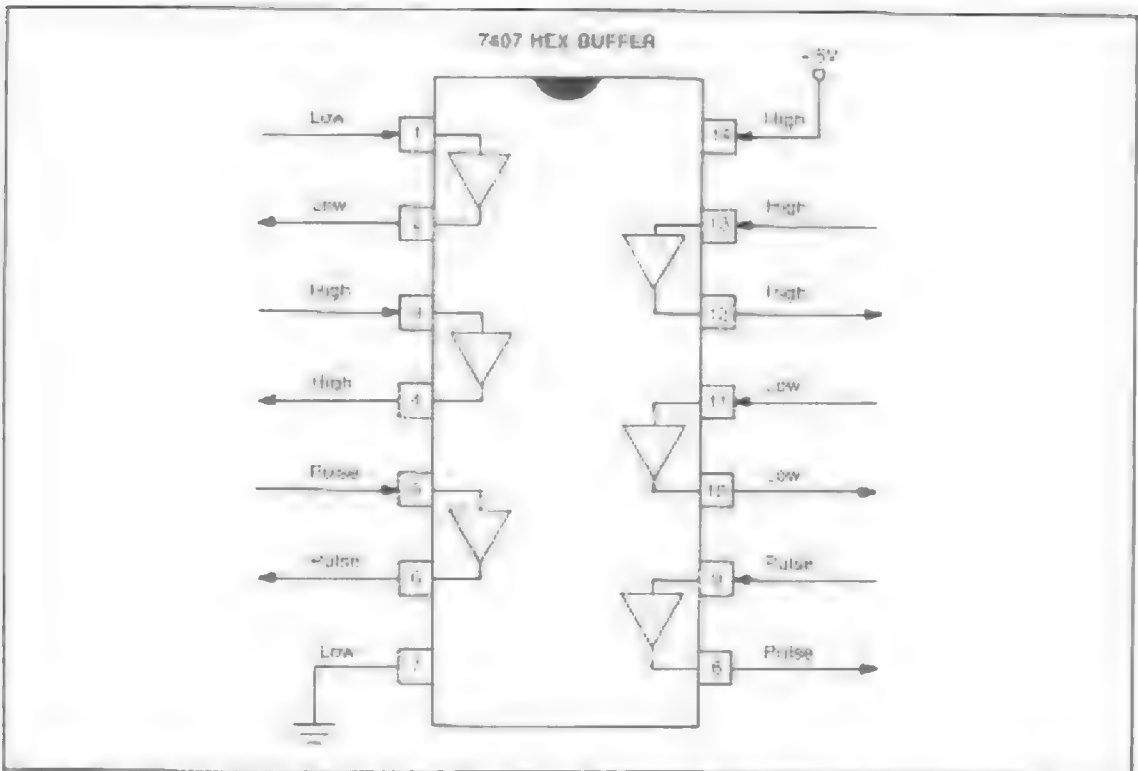


Fig. 8-5. The 7407 Hex Buffer chip has six amplifier stages. Schematically they resemble the NOT gate but without the circle on the output end.

Table 8-2. Two 7407 chips are in the C128. Here are their pin logic states.

Test Point Charts		
PINS	U57	U60
1	Pulse	Pulse
2	Pulse	Low
3	Low	High
4	Low	High
5	Pulse	High
6	Pulse	High
7	Low	Low
8	Low	High
9	Pulse	High
10	High	Pulse
11	High	Pulse
12	Low	Pulse
13	Low	Pulse
14	High	High

in Figs. 8-7 and 8-8. Should one or both of the inputs be low, the AND will not output a high. It simply acts as an open circuit.

Two of the U8 AND gates can be found in the wiring between the two MPUs. Three of the U61 gates are found connected to the AEC and RDY pins of the 8502. The fourth U61 gate is at the AEC pin of the MMU. Note that the AND gates do not have any NOT circles. There are other bullet shaped symbols that look like an AND gate but have circles at their rounded output. They are NAND gates discussed later in this chapter and in Chapter 10.

The other two U8 gates are found by looking for the bullet shapes in the Master Schematic. One is connected to pin 26 of ROM1. The other one is in the 128/64 circuit line. Table 8-3 lists the logic states that should be present.

The AND gates are used as handy control devices to send a high to various terminals to turn

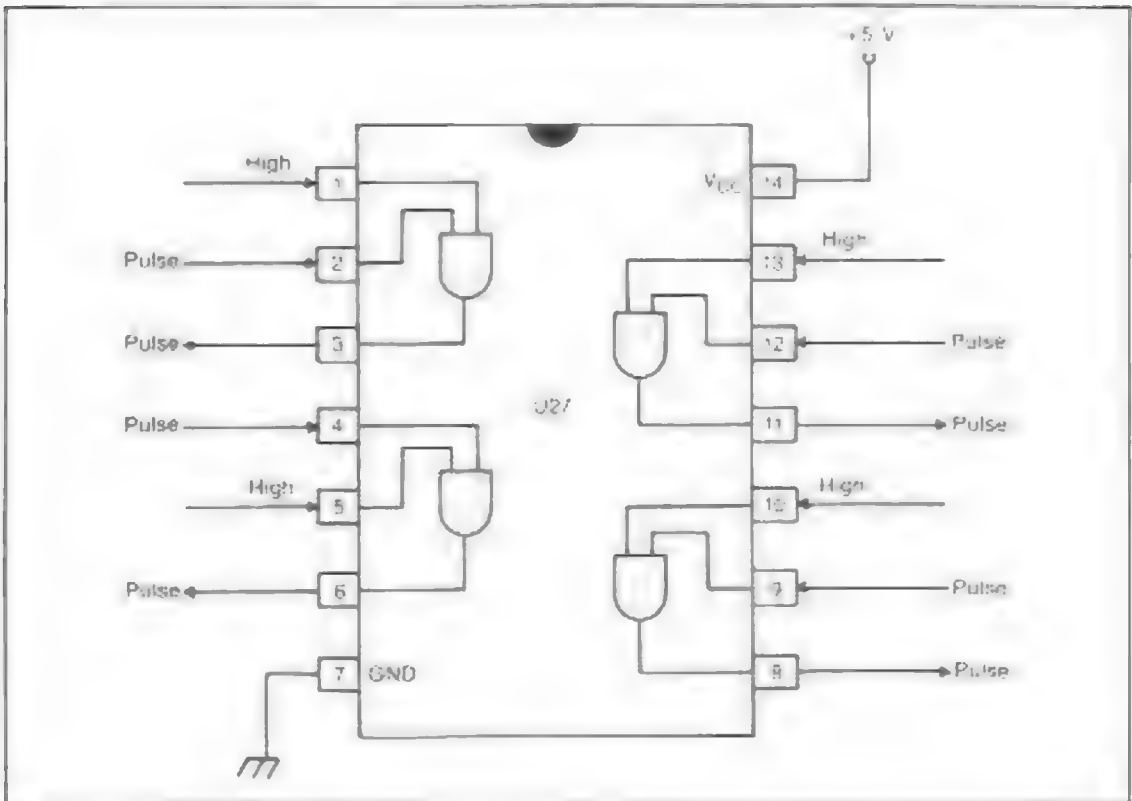


Fig. 8-6 The 74LS08 is a Quad 2-Input AND gate. It has four AND gates on it. Two of these chips are in the C128.

Table 8-3. The logic states on the two 74LS08's are shown.

Test Point Charts		
PINS	U8	U61
1	High	High
2	Pulse	Pulse
3	Pulse	Pulse
4	Pulse	Pulse
5	High	High
6	Pulse	Pulse
7	Low	Low
8	High	Pulse
9	High	Pulse
10	High	High
11	Pulse	High
12	Pulse	High
13	Pulse	High
14	High	High

chips and circuits on and off. A triggering high will be output when two highs are applied to the inputs.

THE 74LS00 & 74LS03 NAND GATES

U31, a 74LS00, in the bottom right-hand side of the printboard; and U58, a 74LS03, slightly above and to the left of U31; are both quad 2-input NAND gates. Both are illustrated in Fig. 8-9. The main difference between the chips is that the 74LS03 has an open-collector output while the 74LS00 does not. Schematically they are drawn the same. When you test them, using Table 8-4, the tests are the same. What is an open collector output?

To begin with, all these 74 - - - type chips are TTLs based on bipolar transistors. Bipolars have collectors as their output electrode. After a gate logically processes the input bits, it is ready to output the finished product through the output collector of

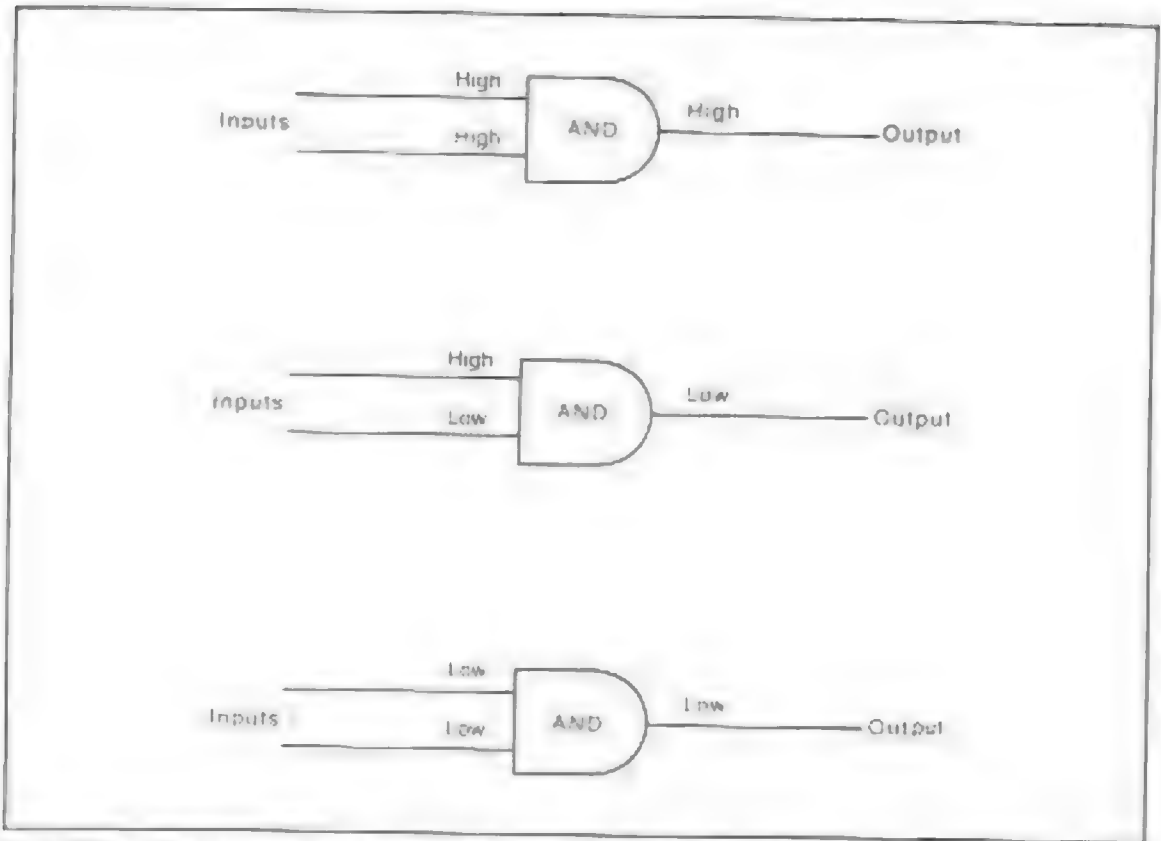


Fig. 8-7. The AND gate should have these three predictable outputs with these inputs.

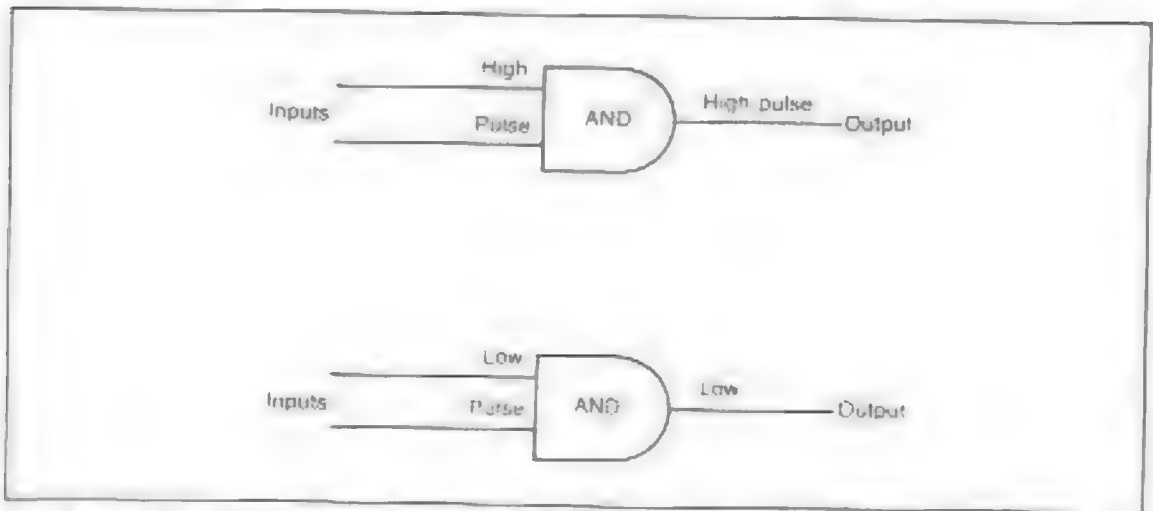


Fig. 8-8. When a pulse is applied with an input high, a high pulse is output every time the input pulse goes high. When a pulse is applied with an input low, the gate never turns on and continually sends out a low.

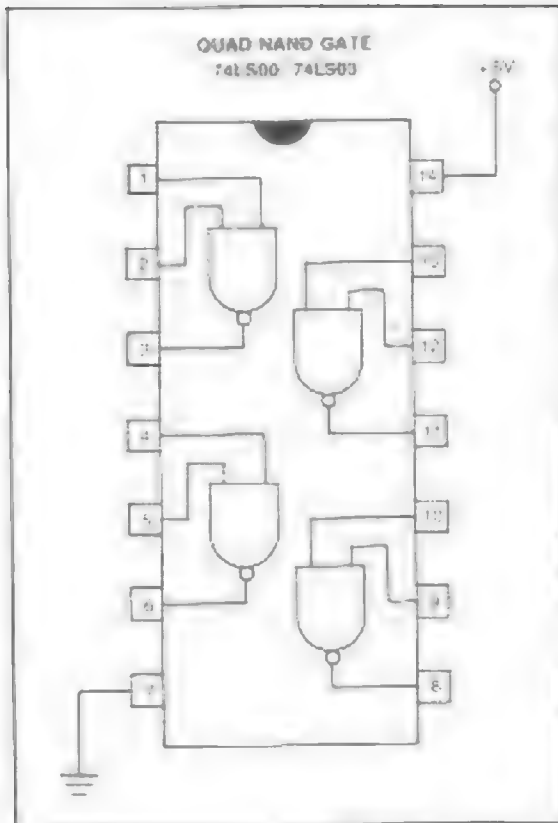


Fig. 8-9. The 74LS00 and the 74LS03 are both NAND gates and drawn schematically in this way.

Table 8-4. The two NAND gates with different numbers, when okay, will probe out with these logic states.

Test Point Charts		
PINS	74LS00, U31	74LS03, U58
1	High	High
2	High	Low
3	Low	High
4	Low	Low
5	High	Low
6	High	High
7	Low	Low
8	Pulse	High
9	Pulse	Low
10	High	High
11	High	High
12	Low	Low
13	Pulse	Low
14	High	High

the stage. With open-collector chips the output transistor is made much heavier than the rest of the tiny transistors in the gate. These output transistors are built to put out a relatively large 30 milliamperes of current.

The internal collector output resistor normally found in a chip that does not feature open-collecting, is left off in the open-collector circuit in chips with the feature. An external discrete pull up or load resistor is installed on the printboard for every open-collector output. That way the chip can handle the extra currents.

If you look on the Master Schematic and find U58—the 74LS03 with the open-collector feature—you'll find pins 6 and 11, two of the outputs, connected to the pull-up resistors, RP2-2 and RP2-4. U31, the 74LS00, without open-collectors, has no pull-up resistors in its output lines. It doesn't need the resistors because the collectors are powered internally. Figure 8-10 shows the way in which the open-collector output transistor is wired.

At first glance the schematic drawing of the quad NAND gates looks like the quad AND gate. A closer look reveals little NOT circles at each output connection. The NOT circles change the AND gate to a NAND, short for NOT-AND. As you read Chapter 10 you'll find that the NAND gate outputs are like the AND gate except that they are NOTed (inverted). For example, the AND gate will only output a high if all inputs are high. The NAND gate will only output a low if all inputs are high. Any other combination of highs and lows at the NAND inputs will output a high.

NAND gates are special. It is much more than the naive implies: an AND gate with an inverter. The NAND gate is the main TTL logic building block. All TTL chips start out as NAND gates. Any desired type of gate or register chip can be constructed from the basic NAND gate. The TTL internal circuits are composed of bipolar transistors, diodes and resistors. There are TTL transistors with multiple emitters, ordinary transistors with single emitters, pn junctions acting as diodes, and tiny integrated circuit resistances.

The TTL emitters are tied to the input pins of a chip. If a gate has two inputs then the TTL has

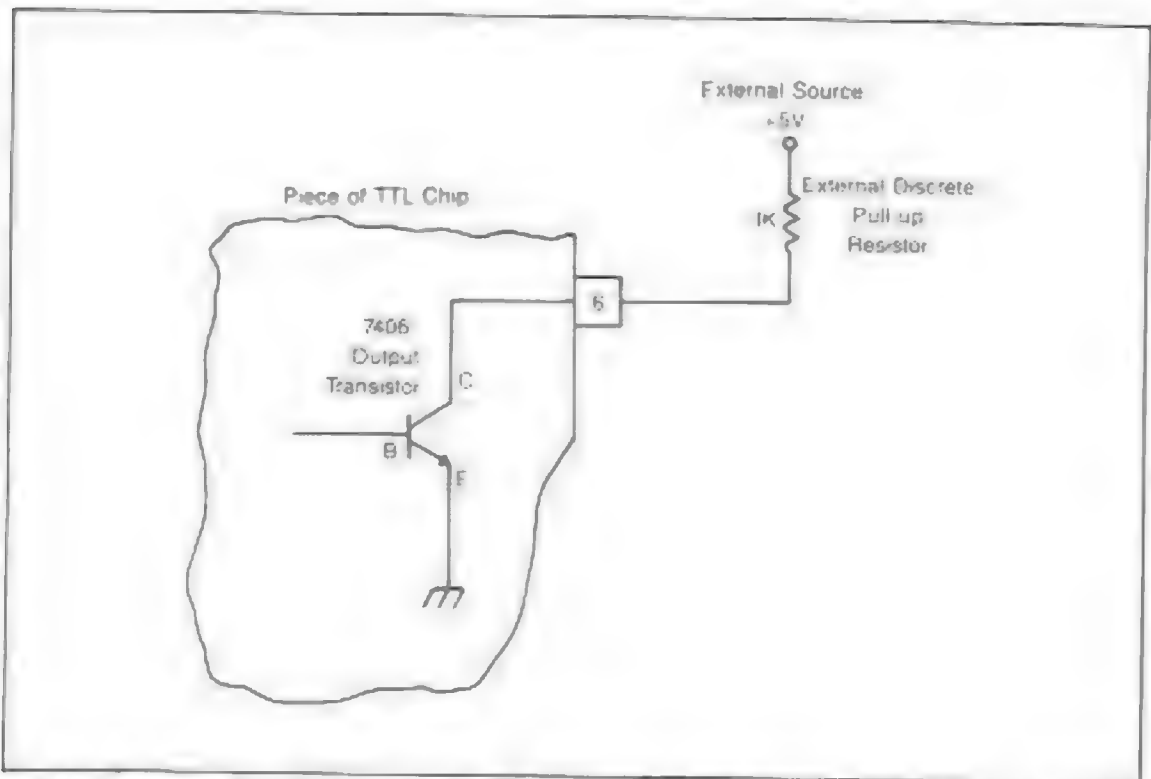


Fig. 8-10 An open collector output means that no pull-up resistors are inside the chip. When such a stage is used, an external discrete resistor must be used. This feature permits the gate to process large amounts of current.

two emitters. Should a gate have three or more inputs, the TTL will provide three or more emitter inputs—whatever is called for. The other bipolar transistors act as inverters or buffers.

The basic NAND circuit that is used in TTLs, shown in Fig. 8-11, consists of five internal circuits. The first circuit is an AND. The AND has multiple emitters. The AND gate will only output a high if the inputs are all highs. The AND circuit output is applied to a NOT circuit. The NOT inverts the AND output. The NOT also amplifies the current of the signal.

The signal is then split in two. One pathway is through a noise immunity buffer, also called a YES gate because it does not change the logic state of the signal. This YES gate outputs to a second YES gate that is built to amplify the signal if it is a high. The second pathway passes the same signal through still another YES gate that is built to amplify the sig-

nal if it is a low. The two final outputs are joined together and output their results. It can be seen that the output ends up as an amplified NANDING of the input signals.

The four NAND gates in the 74LS00 and the other four NAND gates with open-collectors in the 74LS03 are connected to a number of different circuits. Six of them can be recognized because they look exactly like the bullet shaped AND, except they have NOT circles at their outputs. The four U58 sections are all clustered around the serial bus CN6 and the internal serial bus CN7.

Two of the U31 are easily found, one is connected near the MMU-PLA circuits and the other one to the clock and MMU. The other two are not easily found. They are disguised as OR gates (discussed next) with NOT circles at their inputs. This is an alternative way to draw the NAND gates. Figure 8-12 shows the schematic symbols normally

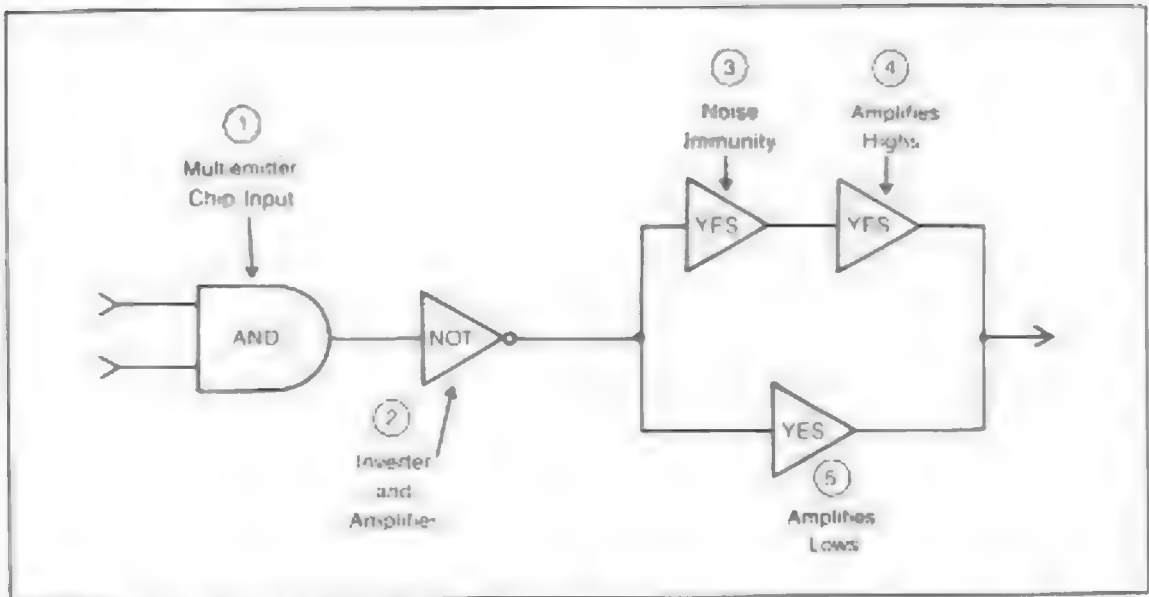


Fig 8-11 The NAND circuit is the basis for all TTL gate chips. The NAND is formed from a multimer AND, a NOT and three YES gates

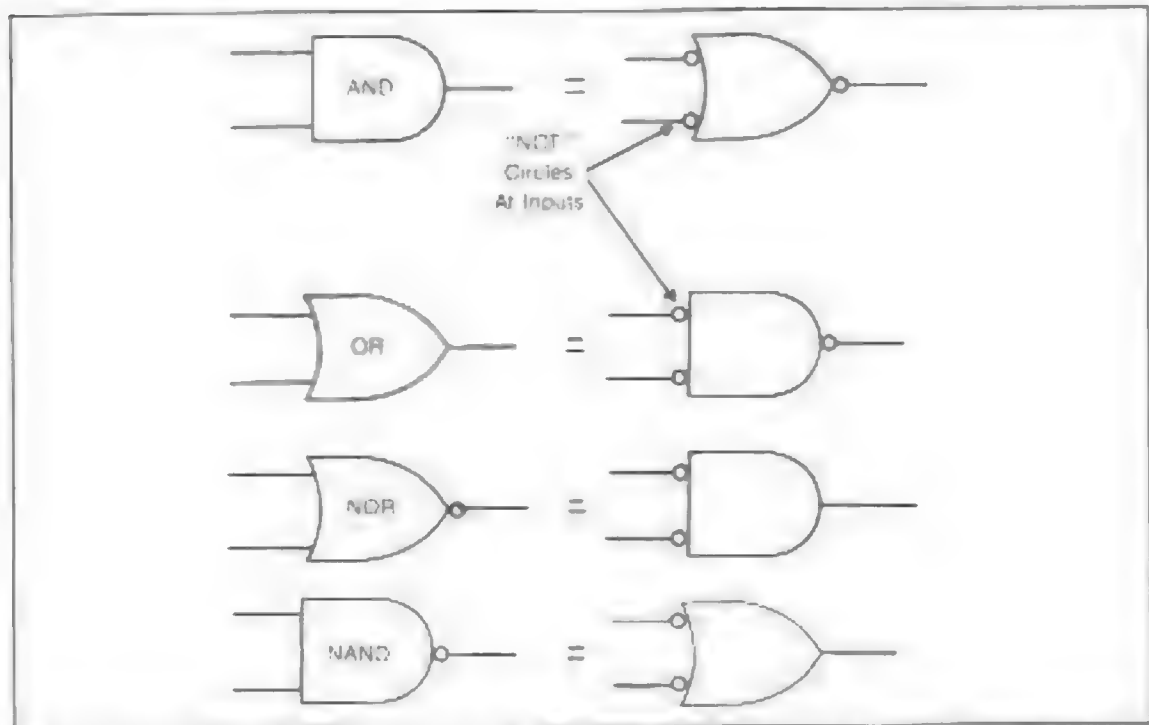


Fig 8-12 Besides drawing gates as shown so far, there are also equivalent sketches. For instance in the C128 clock circuit you will find equivalent NANDs as seen here.

used to depict the various gates. These two differently drawn NAND gates are part of the clock circuit covered in Chapter 17.

THE 74F32 AND 74LS32 QUAD 2-INPUT OR GATES

U9 and U54 both contain four OR gates. U9 is located between the 128K RAM set and the PLA. It is above the character ROM, U18. U54 is found just to the left of the video metal box in the upper left of the board. From a troubleshooting point of view the two chips can be considered identical even though one has an F and the other has an LS. They are both from the 7432 family. Note the shape of the OR symbol in Fig. 8-13. It is somewhat like the AND except for the pointy output end and the curved input end.

Logical OR is another way that bits can be manipulated. While the AND gate will only output a high if all inputs are high, the OR gate will only output a low, if all inputs are low. It too has two or more inputs that combine to produce a single output. However, if any of the inputs are a high then the output will be a high. Only when all inputs are low can an OR gate output a low.

If you look on the schematic for the U9 OR symbols they might be hard to find. They are drawn in the alternative way: as AND symbols with NOT circles at their inputs and outputs. Two of U9's symbols are found at the inputs of the two 64K RAM sets. The other two U9 symbols are at the output of the PLA. The two at the PLA output each input into one of the inputs of the two at the RAM inputs. The OR gates are aiding in the choosing of which RAM sets are to be used.

Two of the U54 OR gates are in the video controller output circuit that leads through U24 a driver stage covered later in this chapter—to the RGBI output port. U54 ORs the Monochrome output of the port.

The 7432 family of OR chips are also 14-pin DIPs housing four gates like the 7408 AND chip family. The pin layouts are identical with the power, inputs and outputs all on the same pins. If you are doing a lot of testing on these pins it would be use-

ful to memorize the power, input and output numbers. The logic states on the pins of U9 and U54 are in Table 8-5.

The NOT, AND and OR gates are the three main logical functions in digital electronics. Others, such as NAND, NOR, Exclusive OR and Exclusive NOR, are all covered in Chapter 10. The main three NOT, AND and OR are the basic gates. These three can be combined to form any of the logical gates. If you combine NOT and AND, then you have a legitimate NAND. Should you put together a NOT and OR, a NOR is created. The exclusive type gates can be produced by wiring NAND gates together and

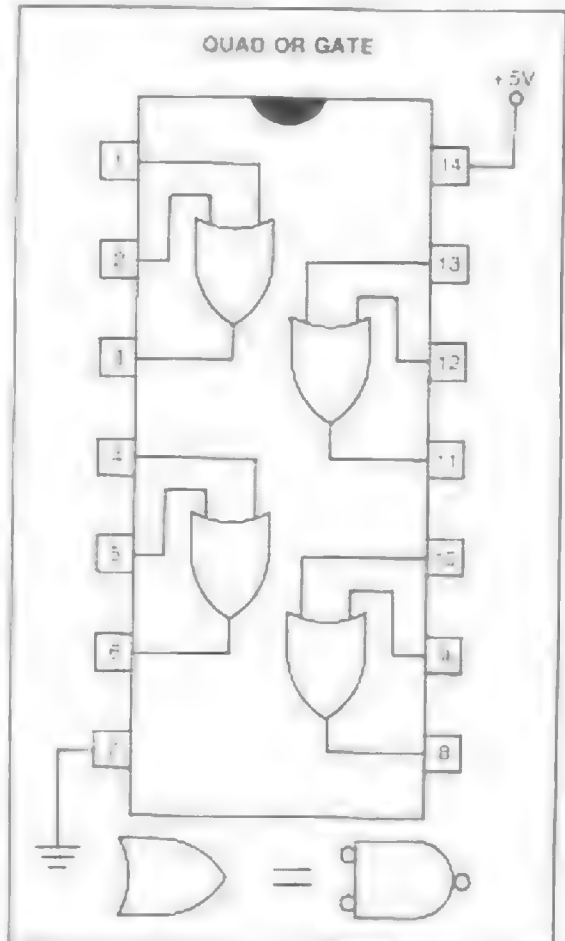


Fig. 8-13. The OR gates look something like ANDs but with pointy ends.

Table 8-5. The two OR gates have the following logic states on their pins.

Test Point Charts		
PINS	74F32, U9	74LS32, U54
1	Pulse	Low
2	Pulse	Low
3	Pulse	Low
4	Pulse	Pulse
5	High	Low
6	High	Pulse
7	Low	Low
8	Pulse	Low
9	Pulse	Low
10	Pulse	Low
11	High	Pulse
12	Pulse	Pulse
13	High	Pulse
14	High	High

placing a NOT gate where needed to invert the logic state.

When you are troubleshooting, though, you do not pay much attention to the overall gate that is created by wiring a number of individual gates together. You simply test individual gates on a chip.

THE 74LS74 DUAL D FLIP-FLOP

The clock circuit of the C128 uses a 74LS74 chip: U56. The clock is discussed in detail in Chapter 17. The flip-flop acts as a momentary storage device in the clock. It is a 14-pin DIP, illustrated in Fig. 8-14. It is found in the bottom right side of the board in a group of five small chips, between U8 and U31. U56 contains two flip-flop circuits. A flip-flop is a storage circuit, unlike gates that simply manipulate logic states. A flip-flop stores logic states. The storage in static RAM is accomplished in flip-flops.

The flip-flops in the 74LS74 though are there to store a state for only a fraction of a microsecond

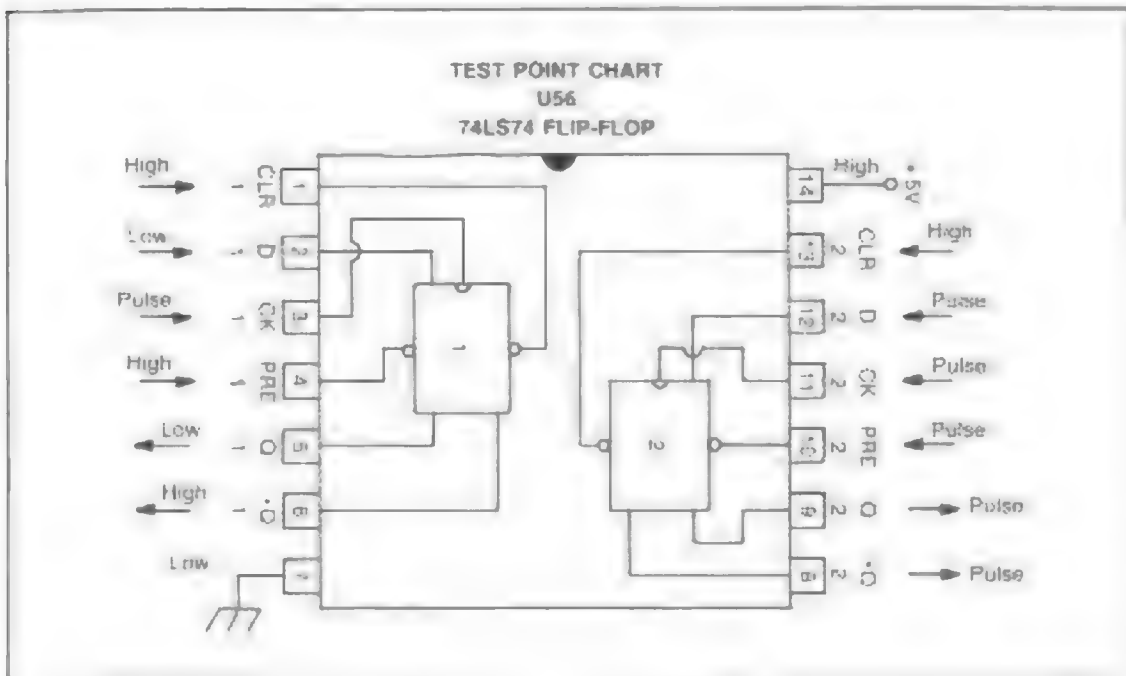


Fig. 8-14 Flip-flops are logic state storage devices.

and then change states as it flip-flops. More detail is given on this circuit activity in Chapter 17. Figure 8-14 shows the logic states of the chip. There are six connections for each flip-flop and two terminals, 7 and 14 for ground and supply voltage.

THE 74LS138 1-OF-8 DECODER

U3 is a 16-pin DIP that is located just to the left of the 8502 MPU. It receives three address lines: A11, A9 and A8 from the 8502. These three address lines are able to form eight different combinations. The chip is a 1-of-8 decoder. According to what bits enter the address line inputs, 1 of 8 output lines emit a bit. The bit that is emitted travels to a chip-select pin of chips like the two CIAs, and selects one for operation.

The chip acts like an assistant to the PLA. The chips that the PLA does not have the capacity to select, the 74LS138 decoder chip will.

The Master Schematic shows a schematic closeup of U3. Note that A11, A9 and A8 enter the chip at pins 3, 2 and 1. The names of the pins are A, B and C. Figure 8-15 is the schematic of the internal wiring of U3. Not shown are the actual transistors, diodes, etc. that each gate is composed of. You can forget about the actual components in each gate and consider the gates as the parts the chip is made up of.

The decoder works in the following way. The three input address lines are able to receive one of eight combinations of bits. They are LLL, LLH, LHL, LHH, HLL, HLH, HHL and HHH. Each one of these input possibilities can be decoded in the internal gates so that one of the eight output lines will put out a resultant bit. The eight output pins are named Y7 through Y0.

The 74LS138 is a 16-pin DIP. Besides the three input address lines and the eight output select lines, there are three input enable pins called G1, G2A and G2B. The usual +5 volt input and ground are at their pin stations and power all the circuits in the chip at the same time.

U3 works through eight NAND gates. Each NAND gate outputs to one pin. For example, NAND

number 0 outputs to pin Y0 and NAND number 7 outputs to pin Y7. Each NAND gate has four inputs. The NAND inputs are coming from the three addressing and three enable inputs. The three enable pins, G1, G2A and G2B are wired to an AND gate.

Let's examine what happens in this AND gate. First of all, the output of the AND gate connects to one of the four NAND input lines on all eight NAND gates. In order for the enable circuit to be able to turn on the NAND gates, the enable AND gate must output a high. If the AND gate outputs a low then none of the NAND gates will operate and the chip will lie dormant.

G1 input is connected to +5 volts which is a fixed high. The high enters the chip and encounters a NOT gate. The high is inverted to a low. It then runs into a NOT circle, however, and is made a high again. The high then enters the AND gate and keeps that AND input in a high state.

The G2A chip input is coming from the PLA, pin 38. This PLA signal is called *IOCS which is a low (note the *). As this low enters G2A, the first component it meets is a NOT circle. The NOT inverts the low to a high. The high enters the AND gate. The G2B input pin is coming from U31, a NAND gate.

The NAND has two inputs. One from address line A10 and the other from the Z80 MPU pin called *M1. When U31 receives two highs at its inputs, it will output a low. That low enters G2B and finds it is at another NOT circle. The low is inverted to a high and enters the AND gate. With three input highs, the AND gate will then output a high and pass a high enabling signal to the eight NAND gates of the chip. The chip will then be able to operate.

Meanwhile, down at the address lines, one combination of three bits is entering. These bits now are in contact with a pair of series NOT gates. As each bit passes through two NOT gates, it ends up being inverted twice in the same state it began. Note that in the output of each series, NOT gates are connected to four of the eight output NAND gates.

Each two series NOT gates are connected. An address bit that is tapped off in this way, between two NOT gates, has its state inverted. Therefore,

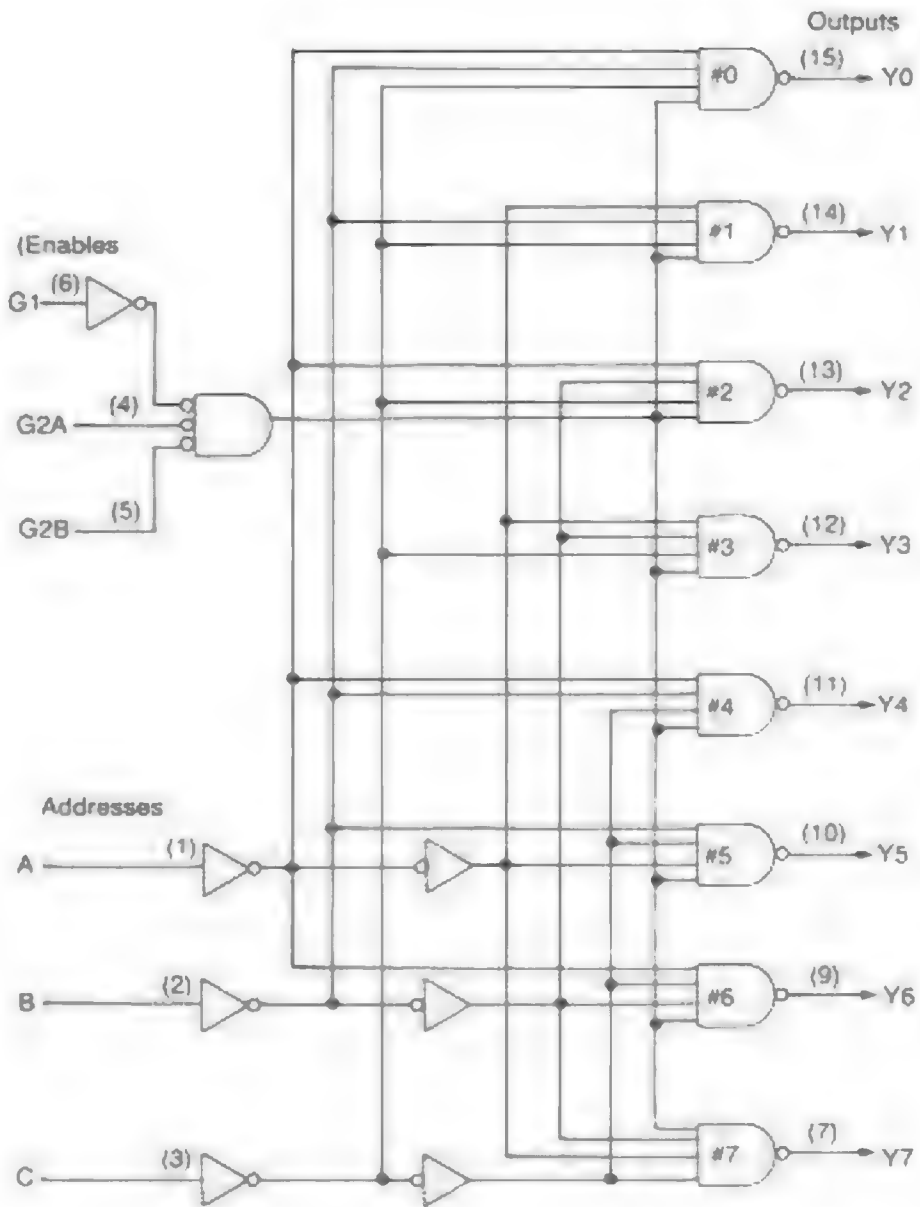


Fig. 6-15 The three enable bits are able to turn on the AND gate which in turn enables one connection on each of the NAND gates. The three address bits then select one of the NAND gates according to the combination of highs and lows

each one of these tapped lines contains the inverted state of the address bit input. Note that each one of these lines is connected directly to four of the eight NAND gates.

Each NAND gate thus receives four inputs. One input is from the enable AND gate and three inputs are from the address line circuits. A NAND can turn on and output a low when all four of its inputs are high. Table 8-6 is called a Truth Table. A way to describe a high is by calling it a TRUE. In the same way a low is called a FALSE. The truth table therefore is nothing more than a listing of inputs, and then the outputs resulting from the inputs. A truth table is the listing of the highs and lows involved. Table 8-6 shows which NAND gate will turn on when U3 is enabled and receives three bits from the address lines.

Let's follow one set of inputs to see which NAND will turn on and output a low, using Fig. 8-15. When a NAND is not turned on, it will simply hold a high state at its output. For example, suppose inputs C, B and A are LHL. (The three enables in this case must be HLL during operation.) When the low at C pin arrives, it then passes through a NOT gate and becomes a high. The high is first tapped off between the two NOT gates and sent to NAND gates 3, 2, 1 and 0.

The high also passes through the second NOT gate, becomes inverted to a low, and is applied to

NAND gates 7, 6, 5 and 4. This low shuts down the operation of these four NAND gates. The other four NAND gates 3, 2, 1 and 0 now have two highs applied to their four input lines. One high is from the enable circuit and one high is from the C address input line.

At the same time, a high is arriving at the B address input. The high passes through a NOT gate and becomes a low. The low is tapped off and sent to NANDs 5, 4 and 1. The low turns them off, too. However, the low also continues straight ahead and passes through the second NAND gate in the B line and becomes a high. The high is then wired into NAND gates 7, 6, 3 and 2. Up to this point, all the NAND gates except 3 and 2 have received at least one low which has turned them off. Only 3 and 2 are left. The bit that enters through the A address line will choose one of the remaining two gates to be the final output. Only one out of the eight NANDs will be permitted to output a chip selection bit.

Entering the A line is a low. It passes through a NOT gate and becomes a high. The high is applied to NANDs 6, 4, 2 and 0. The only one of these four output candidates without any lows is number 2. With this additional high applied, number 2 now has four highs applied. One high is from the enabling circuit, one from the C address line, one from the B line and one is from the A line. Therefore number 2 NAND will turn on and output a low, accord-

Table 8-6. Truth Table for the 74LS138.

Inputs			Outputs, *0 (held high)										
*E1	*E2	E3	A0	A1	A2	0	1	2	3	4	5	6	7
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

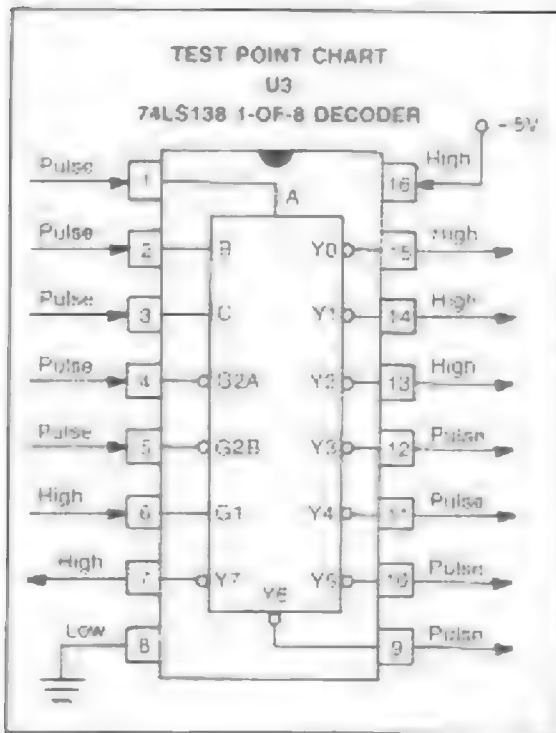


Fig 8-16. The decoder should show these logic states

ing to logic. All the rest of the NANDs have at least one low applied, do not turn on, and their output pins remain held high. In the C128, when number 2 outputs a low, it then becomes a signal called \bullet CS8563. The 8563 chip is the 80-column Video Controller. It enters the 8563 at pin 7 which is called \bullet CS. This is the chip-select pin for the 8563. The 8563 is discussed in Chapter 21.

The \bullet CS8563 low signal is 1-of-8 signals the inputs to U3 can produce. The other signals are \bullet I/O2, \bullet I/O1, \bullet CIA2, \bullet CIA1 and \bullet SID. If you want to practice understanding this decoder chip, then you could trace all the input bit trios and see how each 3-bit address will choose one of the output signals. Figure 8-16 is the Test Point Chart for the U3 chip.

THE 74LS257 MULTIPLEXERS

The C128 uses three 74LS257 chips to do multiplexing jobs. U14 and U15 are found just to the right of the 128K RAM chips. They work with the

two banks of 64K RAM. U26, the third multiplexer, is located two chips to the right of U14. It operates with the VIC.

These multiplexers are 16-pin DIPs and are called Quad 2-Input Multiplexers with three-state capability. The three-state function means that the four individual multiplexer sections on each chip can be completely put into a standby three-state condition with a single input bit. Each chip is able to address eight lines. Two of these chips can control the 16 address lines from the 8-bit 8502 MPU. These are U14 and 15 in the C128. The two multiplexers work with 64K of RAM at a time. Whichever RAM bank is chosen at a particular moment, the multiplexers handle the addressing chores for it.

In Chapter 6 the 4164 RAM chips were covered. They are 16-pin DIPs, and eight of the pins are used for incoming addresses. These eight address lines are given the job of handling 16 address lines, A15-A0, from the MPU. The way the 4164's handle 16 address lines is by accepting only eight at a time. The multiplexers take care of sending eight address lines at a time. This is what multiplexing is.

Figure 8-17 displays how the two 74LS257 chips accept 16 address lines and output to a 64K DRAM set eight lines at a time. Figure 8-18 shows the gates in a 74LS257. There are four input-output circuits in the chip. Each circuit contains two AND gates and one OR gate. Each of the AND gates in a section has one input coming from an address line. This connects eight address lines to the four circuits.

Let's examine one section with two AND and one OR gate. Each AND has one input from an address line and a second input from a chip select pin. In the chip select line there are two NOT gates in series. There are two outputs from the select line one from a tap between the two NOT gates and the second after the two NOT gates. The tap line is connected to four of the AND gates and the other line to the remaining four NOT gates. This makes the two select line outputs have opposite logic states. That way, when one select output is high, then the other is low, and vice versa. Therefore, one will select while its opposite will not. In the C128 you'll find pin 1 is the select and is called \bullet SELA.

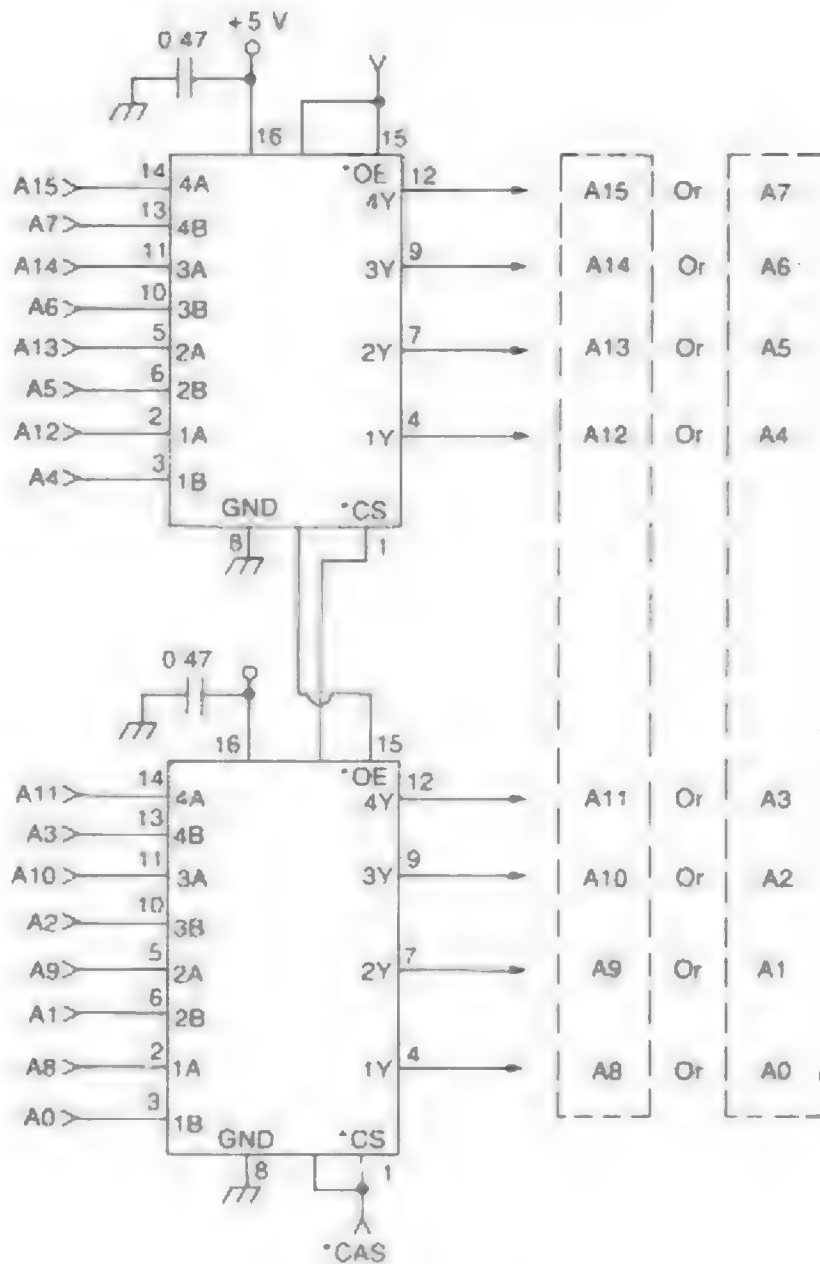


Fig. 8-17 Two 74LS257 multiplex chips receive 16 address lines. They output eight address lines at a time. Eight bits address DRAM row locations and the other eight bits address the column locations.

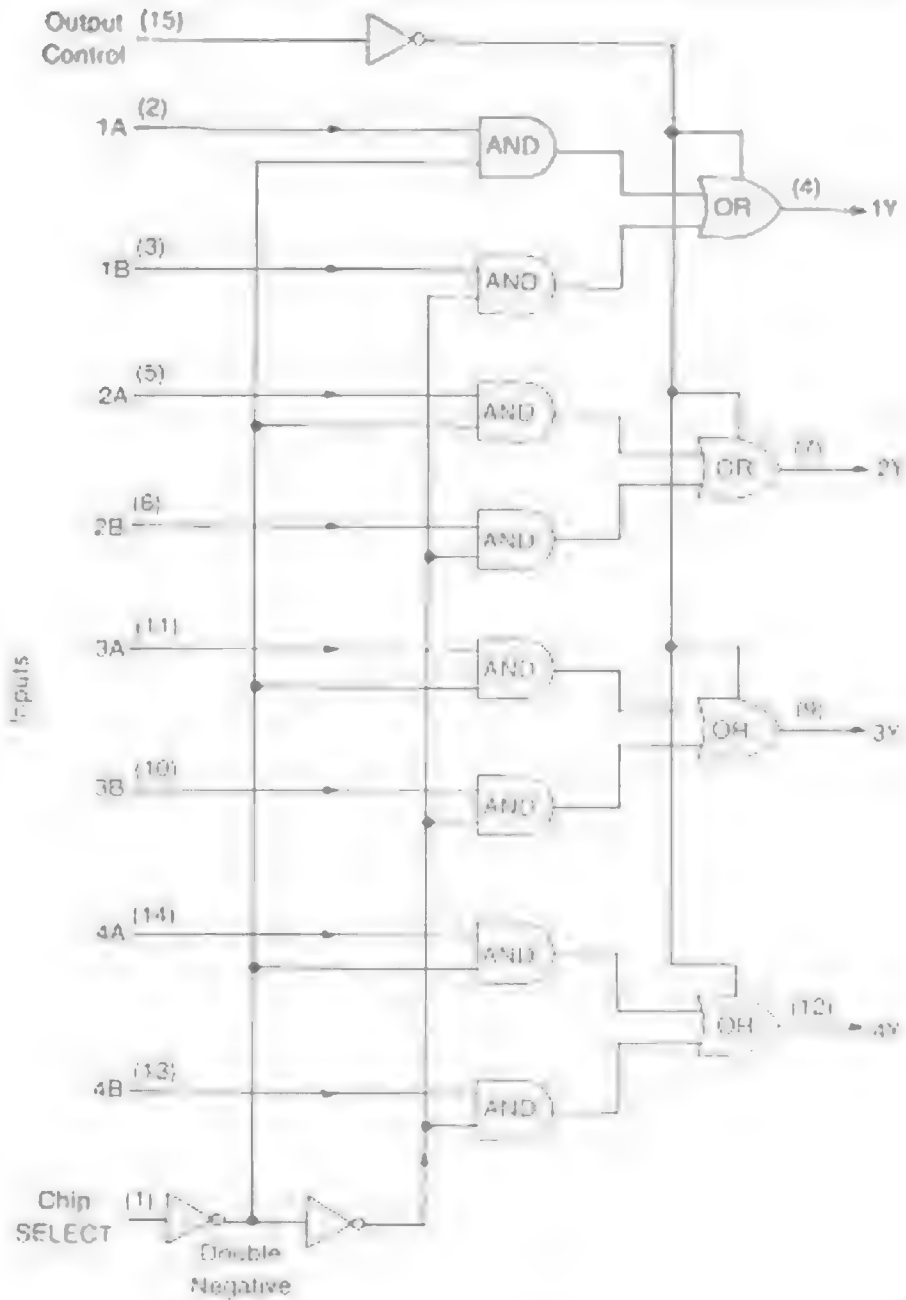


Fig. 5-18 Inside a 2-to-4 multiplex chip are eight AND gates to receive eight address bits and four OR gates to output four multiplexed bits. It takes two chips to handle the 16 address lines in the C128.

Table 8-7 The 74LS257 Truth Table has the different input-output possibilities.

INPUTS					
Pin 15	Output Control	Select Pin 1	A	B	OUTPUT Y
	H	L	L	L	High Impedance
	L	L	H	L	L
	L	H	L	L	H
	L	H	L	H	L
	L	H	H	L	H

Inside the chip the AND output goes to one of its OR gate inputs. The ORs are three-state and have their three-state connections wired together. The three-state signal enters at pin 15 called •E for enable. It is technically known as the Output Control pin. When a high enters at •E the chip outputs go into a three-state high impedance condition and there is no output from any of the OR gates. The truth table, Table 8-7 shows all of the different input and output conditions.

These multiplexers typically perform in the following way. If you recall in the RAM chapter, the 4164 memory matrix is laid out in a grid of 256 rows and 256 columns. Each row needs eight hits to address 256 locations as does each column. The 16 address lines are connected to the 16 address pins on the two multiplexer chips.

The address hits arrive. In addition, a select bit arrives too. When it is high, four of the AND gates of the eight on the chip are turned on. Because there are two chips, eight AND gates are turned on and eight bits are outputted through the OR gates. These bits could be the ones that address all 256 rows on the DRAM set.

If the select bit is a low, the eight AND gates that were on go off and the ones that were off go on. The other eight bits of the 16 address lines are then passed through the OR gate and address the 256 columns of the 4164 chips that are in operation at that moment. Figure 8-19 is the Test Point Chart for both U14 and U15. Their logic states appear the same on the probe.

THE 74LS373 OCTAL LATCHES

U12, one of the octal latches, is located at the bottom right hand corner of the printed board. U17, the

U12's other octal latch, is just to the left of the PLA. U12 is the latch that operates with the Z80 MPU. The Z80 cannot directly interface with the rest of the digital circuitry. The circuits are designed to work with the 8502 MPU. In order to interface, the Z80 needs the help of U12, and also U13 which is discussed next. U17 is also a 74LS373 and is found in the VIC environment. Both U12 and U17 are discussed again in the Z80 and VIC chapters.

The 74LS373 is a 20-pin DIP. Figure 8-20 is a schematic drawing of the chip. There are eight flip-flops inside the latch. They are said to be "transparent." This means that the output immediately follows the input as long as the chip is enabled. The enabling signal is a high.

A signal passes right through the chip, from an input to an output when pin 11, enable, is high. When enable is low, the chip does the latching. When enable is low, the eight highs and lows that enter the D inputs get latched or stored in the flip-flops. The way that a flip-flop stores a high or low is covered in Chapter 11. When the enable subsequently goes high, the bits that are stored are then transferred to the output pins named Q.

U17, at pin 11 (enable), is strobed by the VIC originated signal •RAS. When •RAS arrives, it unlatches any signal that might be stored in the flip-flops. Pin 1 gives the chip a three-state control. It is connected to the system AEC line. While AEC is held high, the chip three-states, which is also called a high impedance state. The chip simply plays dead. Then when AEC goes low, the chip is awakened and is free to conduct its data transferring duties.

The latch is a staging area for the address bits that enter the eight lowest address lines of the color

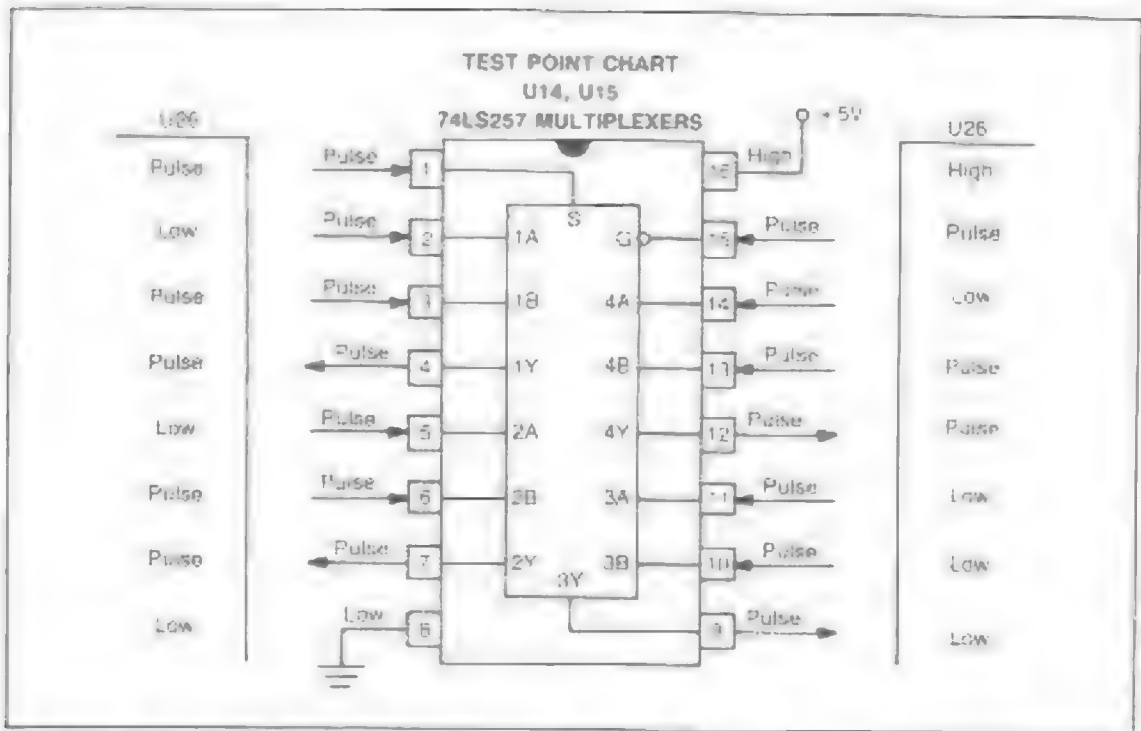


Fig. 8-19 The Test Point Chart shows all pulses except for +5 volts and ground.

KAM. It aids in the choosing of colors for the character blocks in the TV display.

A latch is a simple device. It is able to temporarily store bits as the bits pass through the digital circuits. A latch could almost be thought of as a form of static RAM. Figure 8-21 and its Table 8-8 provide the logic state arrangements when in the C128 mode.

THE 74LS244 OCTAL DRIVERS

The C128 has three octal driver chips with the same generic number: U13 is just above U12, to the right of the Z80 MPU; U24 is in the video box to the right of the 8563; U62 is to the immediate left of the MMU. Figure 8-22 shows the pin layout. Table 8-9 has the logic states.

The chips are 20-pin DIPs. A *driver*, also called a *buffer*, is an amplifier. U57 and U60 are also buffers, but are hex buffers. These chips are octal, which means that they contain eight individual am-

plifiers. The 74LS244 has another feature. It is also an inverter. The chip has eight input lines, 2, 17, 4, 15, 6, 13, 8 and 11. These input lines connect to their assigned buffer, as shown in Fig. 8-22, and then output to their respective pins, 18, 3, 16, 5, 14, 7, 12 and 9. Pins 1 and 19 are inputs for a three-state signal. Each pin controls four buffers. Power is supplied at pin 20 and ground is at pin 10.

U13, as mentioned in the last section, works with latch U12 to interface the Z80 to the 8502 designed circuitry. The interfacing will be covered in more detail in Chapters 12 and 13, but for now here is an overview on what is happening. Refer to the Master Schematic.

In order for the Z80 to use the 8502 circuits it must control the address and data bus lines. They both are going to use the same lines. The address lines are easy to handle. The Z80 has built-in tristate circuits to turn off the address lines. The Z80 can be isolated by simply tristating address lines. The data lines, however, are harder to manage.

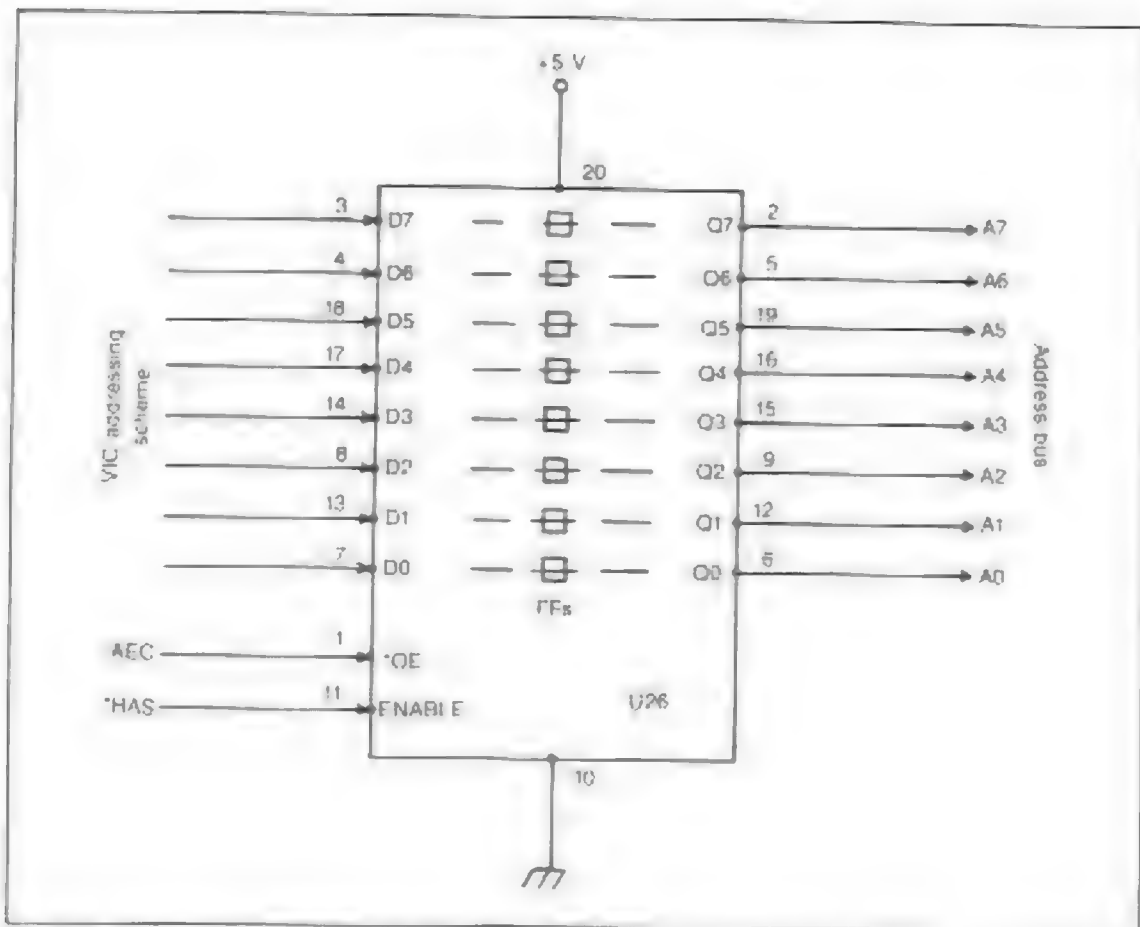


Fig. B-20. The 74LS373 contains eight flip-flops. Each one has a pin for input and output.

That is where the latch U12 and the buffer U13 work. During a read, when data bits are traveling from the memory map to the Z80 over the data bus, the data is latched into U12 and then, when needed, passes on into the data lines of the Z80. During a write, when data bits are traveling from the Z80 to the memory map, the data enters U13. The data is then amplified and passed out to the data bus, D7-D0. Between U12 and U13 the system data bus is interfaced to the Z80. The 8502 has no trouble with the data bus because it was designed for the 8502.

U24, another 74LS244, operates in the 80-column RGBI output lines. Its pins 2, 4, 6 and 8 receive R, G, B and I from the 8563 video controller

chip. Inside U24 these digital video bits are amplified and matched directly to the R, G, B and I pins on the output port.

In addition, taps from the R, G, B and I lines are sent to a network of OR gates, buffers and transistors. These components mix and amplify the RGBI signals and produce one total video signal called *Monochrome*. Monochrome is a black and white video representation that is available as an 80-column output at pin 7 of the output port. This can be fed to a monochrome monitor and will display the 80-column output.

The third octal driver, U62, is sitting next to the MMU and works with eight pins of the MMU. (One of the functions of the MMU is to generate the

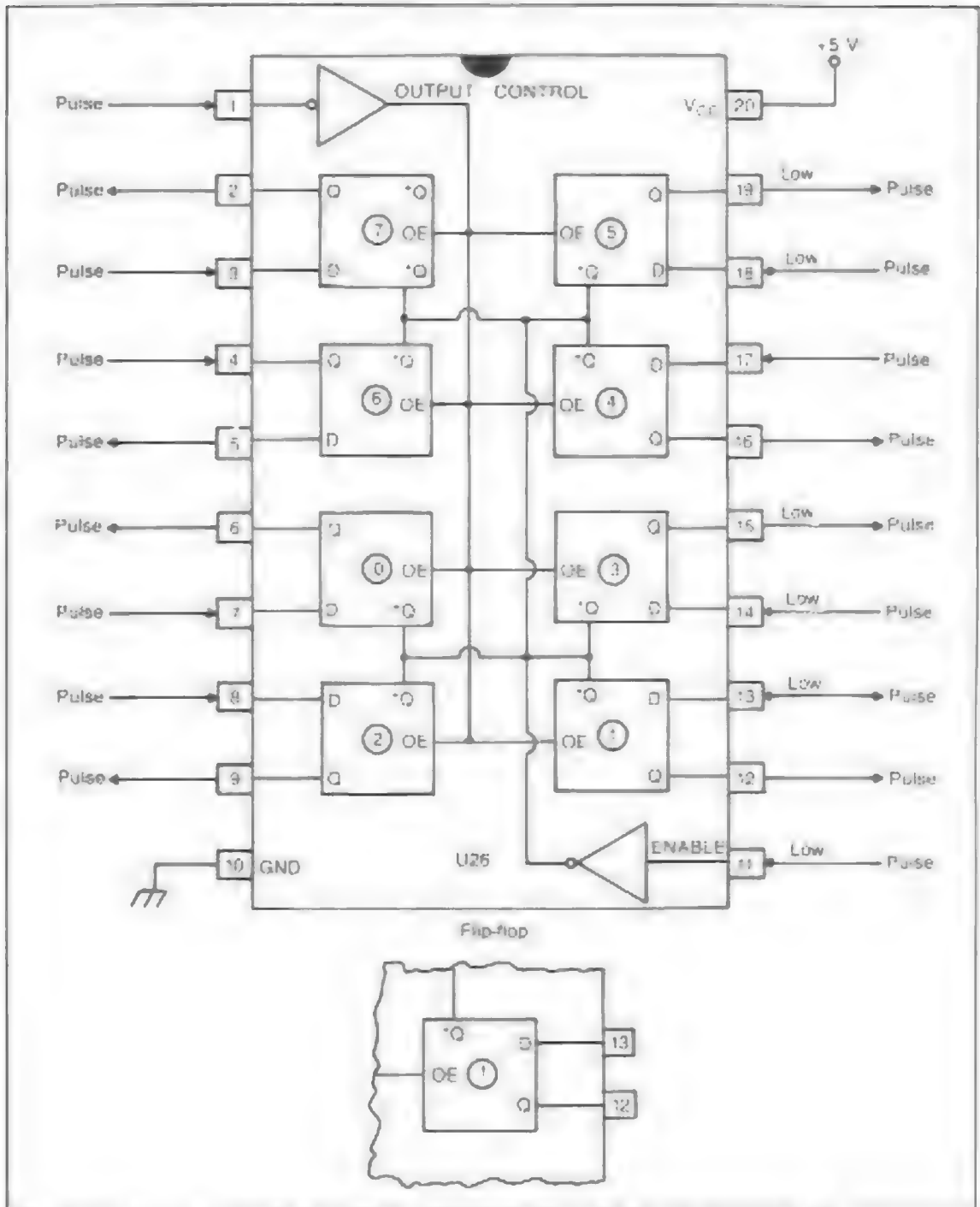


Fig. 8-21. The flip-flops have inputs at Q and D. They are each enabled at \bar{Q} and controlled at OE.

Table 8-8. Two 74LS373 chips are in the C128. These are the logic state readings.

Test Point Charts		
PINS	U12	U17
1	0	Pulse
2	0	Pulse
3	Pulse	Pulse
4	Pulse	Pulse
5	0	Pulse
6	0	Pulse
7	Pulse	Pulse
8	Pulse	Pulse
9	0	Pulse
10	Low	Low
11	Pulse	Pulse
12	0	Pulse
13	Pulse	Pulse
14	Pulse	Pulse
15	0	Pulse
16	0	Pulse
17	Pulse	Pulse
18	Pulse	Pulse
19	0	Pulse
20	High	High

0 - No light

Translated address output lines. There are eight of them called TA8-TA15. They exit the MMU at pins 3-10. These signals are used as an input to the multiplexers instead of the usual A8-A15 lines. This is because they must be tristated in the MMU during the low of a signal called AEC. U62 has the job of receiving TA8-TA15 signals and buffering them. The signals enter the input lines of U62 as TA8-TA15 and exit the output lines as A8-A15.

THE TRANSCEIVER 74F245

U55 is located on the printboard just to the right of ROM3. It is a 20-pin DIP. It is called a transceiver because it is able to receive or send signals from both its inputs and outputs. That is, its inputs can be changed to outputs and its outputs can also be changed to inputs. If you look at Fig. 8-23, the Test Point Chart, you can see there are 16 buffers and

two AND gates. One buffer is used for every input and output line. The signals can travel in both directions like the data bus is able to move signals. The two AND gates are there to tristate the buffer sets so there are no conflicts of signals. One AND gate can tristate signals going in one direction and the other AND gate can tristate signals going in the other direction.

In the C128, pins 2-9 receive address lines A7-A0, see the Master Schematic. Inside the chip these digital bits can be amplified and exit through pins 18-11 in normal fashion.

Besides simply sending A7-A0 on its way, U55 is also able to reverse directions. After A7-A0 passes through U55 it becomes the shared address line: SA7-SA0. It is shared by both the processor and VIC with common circuits like the character ROM and the color RAM. However, during a signal called *DMA that is applied to pin 1, the internal AND gate tristates the forward direction and the reverse direction is opened up. The SA7-SA0 lines force their bits in reverse. This permits external circuits like an expansion cartridge to address the system RAM and ROM. That is why a two way transceiver chip is needed in the circuit.

THE SCHMITT TRIGGER 74LS14

U16 is a Hex Schmitt Trigger chip. On the board it is immediately to the left of the 8502 MPU. It is a 14-pin DIP (see Fig. 8-24) and as the name states has six trigger circuits—all of the inverter type. The symbol of the triggers is like the NOT gate, except that a box-like shape is inside each triangle symbol.

A Schmitt circuit is a circuit that is designed to output square waves. In digital circuits, only square waves are to be used. If a sine wave or some other waveshape should be applied to a digital circuit, then trouble could occur. Therefore, Schmitt trigger circuits are placed strategically in series in digital circuit paths. They receive input signals and ensure that the signals are output from the trigger circuit in a square wave format.

Figure 8-25 is a square wave. It is the wave formed by graphing voltage in the vertical axis and frequency or time on the horizontal. The waveshape

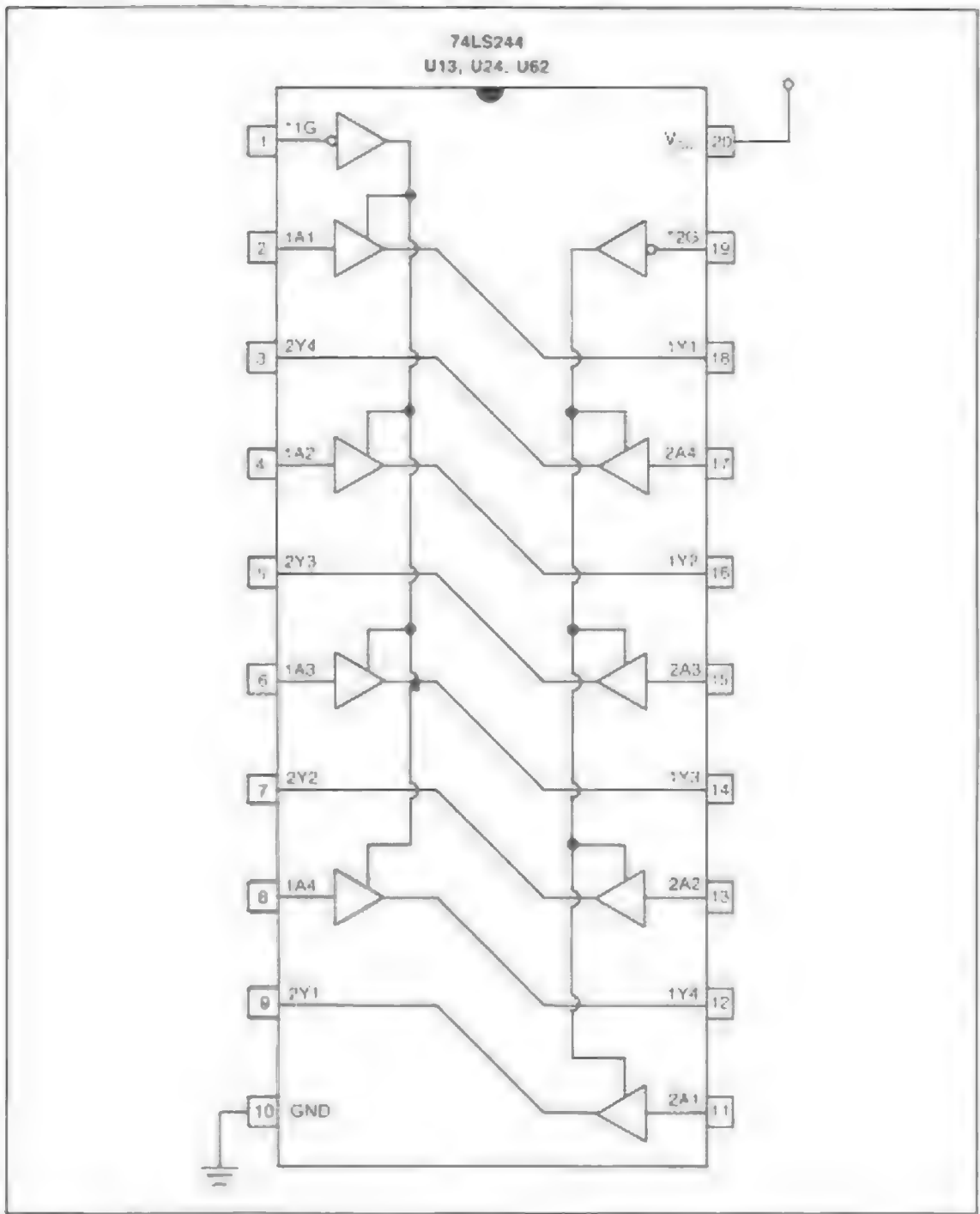


Fig. 8-22 The three 74LS244's all have this pin arrangement

Table 8-8. The logic states on the three 74LS244's are quite different from each other.

Test Point Charts			
PINS	U13	U24	U62
1	High	Low	High
2	0	Low	Pulse
3	Pulse	Low	Pulse
4	0	Low	Pulse
5	Pulse	Pulse	Pulse
6	0	Low	Pulse
7	Pulse	Low	Pulse
8	0	Low	Pulse
9	Pulse	Pulse	Pulse
10	Low	Low	Low
11	0	Pulse	Pulse
12	Pulse	Low	Pulse
13	0	Low	Pulse
14	Pulse	Low	Pulse
15	0	Pulse	Pulse
16	Pulse	Pulse	Pulse
17	0	Low	Pulse
18	Pulse	Low	Pulse
19	High	Low	High
20	High	High	High

0 = No light

has names for its parts. The wave, if viewed on a scope, traces out the shape in this way. The first vertical line is called the *rising edge*. The top horizontal line is the *high*. The second vertical line is the *falling edge*. The bottom horizontal line is the *low*. Note that the high is at +5 volts and the low is at 0 volts.

In the computer all of the signals that are traveling through the digital circuits are forms of square waves. Actual rising and falling edges are not straight up and straight down. That would mean the charges were taking place in no time at all. The rising edge actually slopes somewhat to the right and the falling edge to the left.

As long as the slopes are slight, the durations are flat and the slopes and durations are joined in a sharp square point, the waveshape can be processed satisfactorily in the digital circuits. Should the

waveshape in question be a slowly varying irregular pulse, then the digital circuits cannot handle it. That is where the 74LS14 comes in. It will take a poorly defined pulse and output a square wave.

In the C128 Schmitt triggers are installed in the following places, and are shown on the Master Schematic. First of all there is a trigger in the 9 volt ac line that connects to the TOD pins on the two CIAs. The trigger simply takes the 60 Hz sine wave and changes it to a 60 Hz square wave. The 60 Hz originates at the electric company. The TOD pin is connected to the Time Of Day circuit in the CIAs. The square wave locks the TOD circuit in time with the electric company so that the computer's clock will run precisely on time.

Two more Schmitt triggers are placed at the internal serial port circuits. The triggers both output into one input of two parts of U58—a quad NAND. The triggers ensure that the serial bus is only moving square waves and not other ill-defined pulses.

Two more triggers from U16 are installed in the RESET and RESTORE circuits. They are in the lines that lead to two NOT gates from U37 and U30. All of these triggers are installed in circuits that are on the fringe of the digital circuits. At these places the square waves are in contact with other types of wave and must be clearly defined before allowing them into the digital world. Normally triggers are not needed once the square waves are deep into digital territory.

THE 4066 QUAD BILATERAL SWITCH

There are two 4066's in the C128. U2 is located on the printboard next to and on the right of the MMU. U20 is found next to and on the right of the VIC but outside of the video box. U2 is connected to the two control ports and inputs to SID. U20 is installed between the color RAM and VIC. It handles the color signal that VIC reads out of RAM.

As its name implies, there are four sections to the 4066 and the sections are switches, shown in Fig. 8-26. They are four on/off switches, electronically controlled. They are handy to control data type buses or select data in other ways. Each of the four switches have three connections. There are, first of all, two I/O lines. Signal can flow in these lines

TEST POINT CHART
U55
74F245 TRANSCEIVER

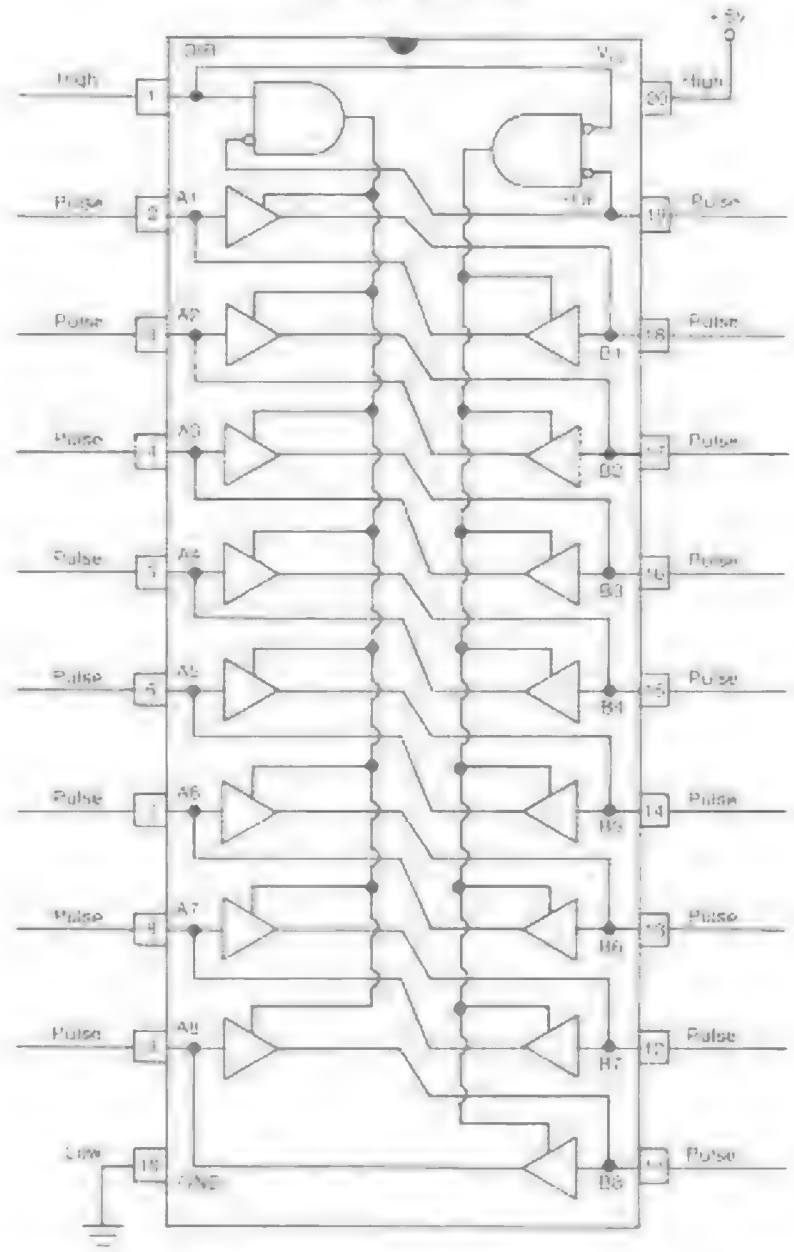


Fig. 8-23 The 74F245 is an amplifier chip that can send signals in two directions.

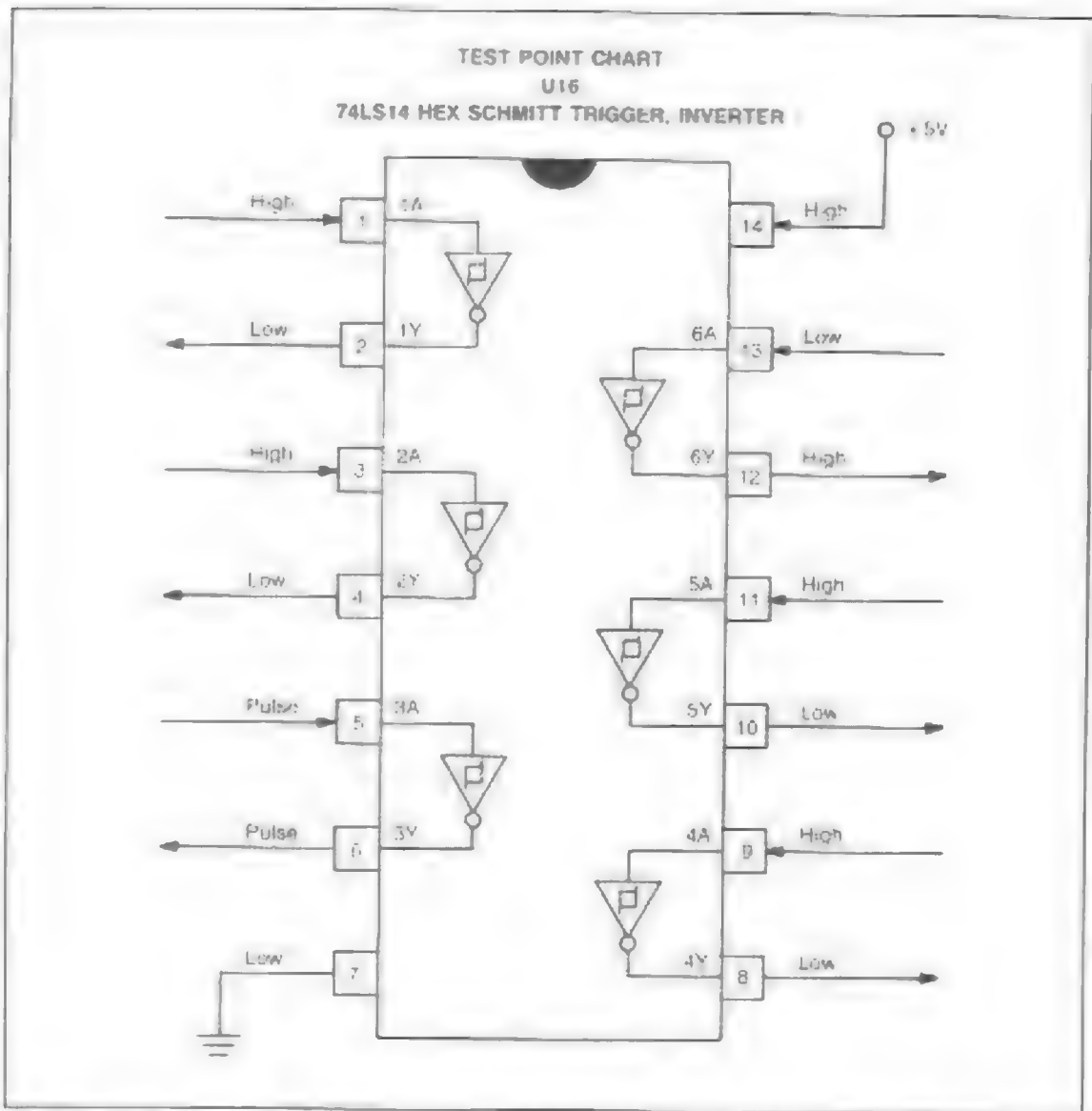


Fig. 8-24 The 74LS14 is a device that can change most waveforms like sine waves to square waves.

either way. The I/O lines for the A section are pins 1 and 2. B section are pins 3 and 4. C section are pins 8 and 9 while D section are 10 and 11. Nothing is mysterious—these are off-on connections.

The third connection is the off-on control. When you want to close a switch, you connect its control connection to a high. If you want it permanently

closed, then attach it to the supply voltage, pin 14, that is supplied with a steady +5 volts—a high.

If you want to open a switch, then connect its control connection to a low. Should you want a permanently open switch, then connect the control pin to the ground of the chip, pin 7. This low puts 0 volts on the control and the switch will not close.

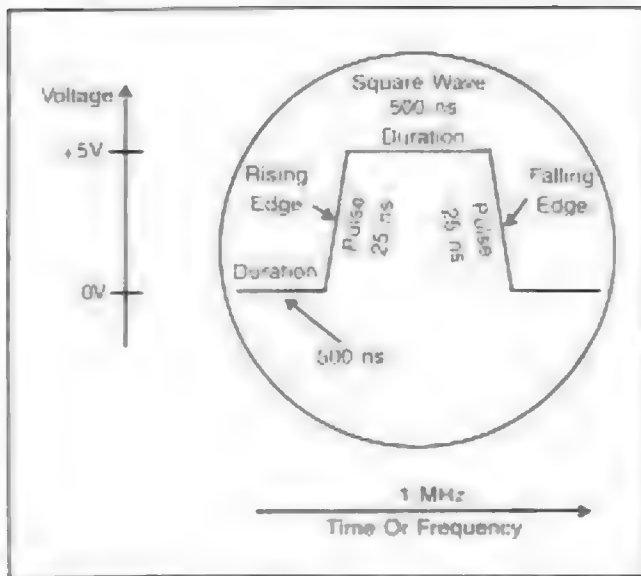


Fig. 8-25. The digital square wave is a voltage level that passes through time and periodically changes levels from 0 volts to +5 volts and back.

The control pin for the A switch is pin 13. The control pin for the B switch is 5, the C switch is 6 and the D switch is 12.

These 4066 switches are fairly rugged. They can take a supply voltage up to +5 volts. The C128 only uses +5 volts. The only problem with switches in sensitive digital circuits is the static electricity that can build as the chips do their switching. If the 4066 does fail, it is usually due to this problem. They can be tested quickly with a logic probe or vom using the 4066 Test Point Chart, Table 8-10.

THE 556 DUAL TIMER

U27 is the only 556 chip in the C128. It is located on the printboard just below and to the left of SID. As its name implies, two separate timer circuits are on the chip. One of the timers is in the •RESET circuit while the other is in the •NMI arrangement. They are connected as shown in Fig. 8-27.

The timers are used to clock out cycles as the reset or NMI interrupt is controlled. These timers are able to time events ranging from a microsecond to hours. They are versatile devices and used in many applications besides those in the C128.

Each timer is based on a flip-flop storage circuit. The single FF in one timer can flip-flop once

or a large number of times according to the control applied. The control is given through comparator circuits, as illustrated in Fig. 8-28, the Test Point Chart.

When a section of the 556 is made to flip-flop once, it is said to be acting as a *one-shot timer*. If a section is made to keep on flip-flopping, it is called an *unstable oscillator*.

The reset circuit uses one section of the 556 to time out the reset sequence. It uses pins 8, 9, 10, 11, 12 and 13. The reset circuit connects to both the 8502 and Z80 processors. In fact, reset is the only processor control signal that is shared. The reset circuit also connects to the two CIAs and the SID chip.

When the reset button is pressed, control of the system is given to the Z80 and the 8502 is placed on standby. Incidentally, that is the same condition that occurs when you turn on the C128.

The Z80 then goes about initializing and then turns control over to the 8502 in either C128 or C64 mode, according to conditions. Once the 8502 is in operation it turns over control to the Kernel initialization routine called START. START goes about its business of resetting the system.

The other timer in the 556 is wired into the •NMI circuit. If you hit the RESTORE key on the

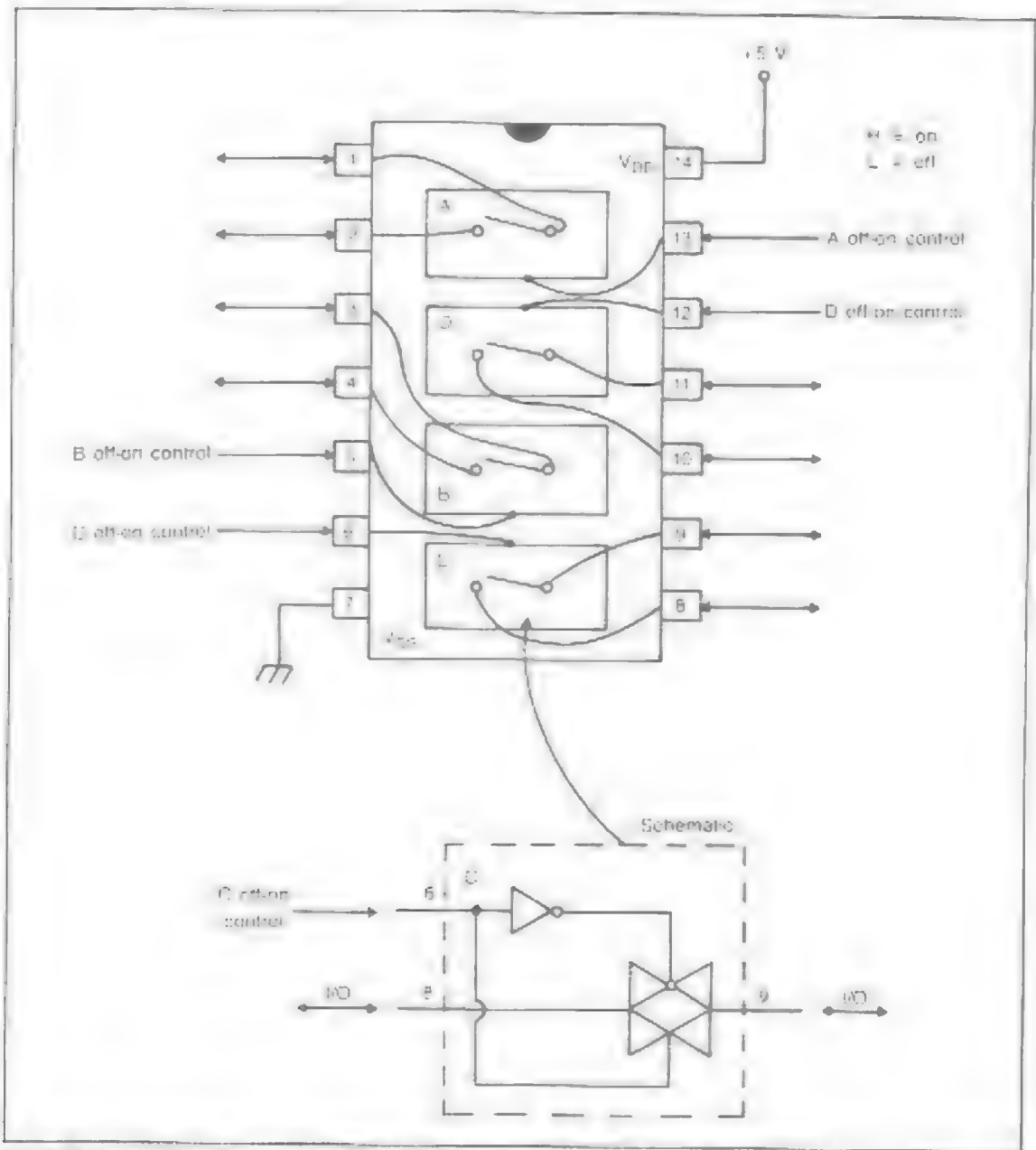


Fig. 8-26 Four electronic switches are in the 8568. Each switch has two input/output pins and an off-on pin.

keyboard then you generate a signal called RSTR. It is applied to pin 6 of the timer. The timer circuit then generates a non-maskable interrupt, *NMI, that is applied to the 8502 pin 4 called *NMI. The

8502 senses the negative edge of the signal. The processor then reads the NMI vector out of memory. The vector is simply an address in ROM where the NMI program routine is located. The processor

Table 8-10. Two 4066 chips are in the C128. These logic states should be on the pins.

Test Point Charts		
PINS	U2	U20
1	0	Pulse
2	Pulse	Pulse
3	Pulse	Pulse
4	0	Pulse
5	Low	Pulse
6	Pulse	Pulse
7	Low	Low
8	Pulse	Pulse
9	Pulse	Pulse
10	Pulse	Pulse
11	Pulse	Pulse
12	Pulse	Pulse
13	Low	Pulse
14	High	High

0 - No Light

then reads the routine and executes it. The routine disables the other interrupts (IRQs), and saves the processor's register contents onto a section of RAM called the *stack*. The processor then continues through the NMI sequence according to what is needed for the RESTORE effect. There will be more about the •NMI in Chapter 12.

OTHER CHIP-LIKE COMPONENTS

Of the 63 U components in the C128, the last four chapters discussed 61. There are two more, U28 and U59. U28 is located in the video box to the left of VIC. U59 is not a DIP. U59 is a transistor device to the right of the MMU.

U28 is in a 16-pin DIP. It is numbered 8701 and controls the frequency of the master oscillator of the C128. The master oscillator runs at a crystal-controlled frequency of 14.31818 MHz. This frequency is needed to derive the various frequencies

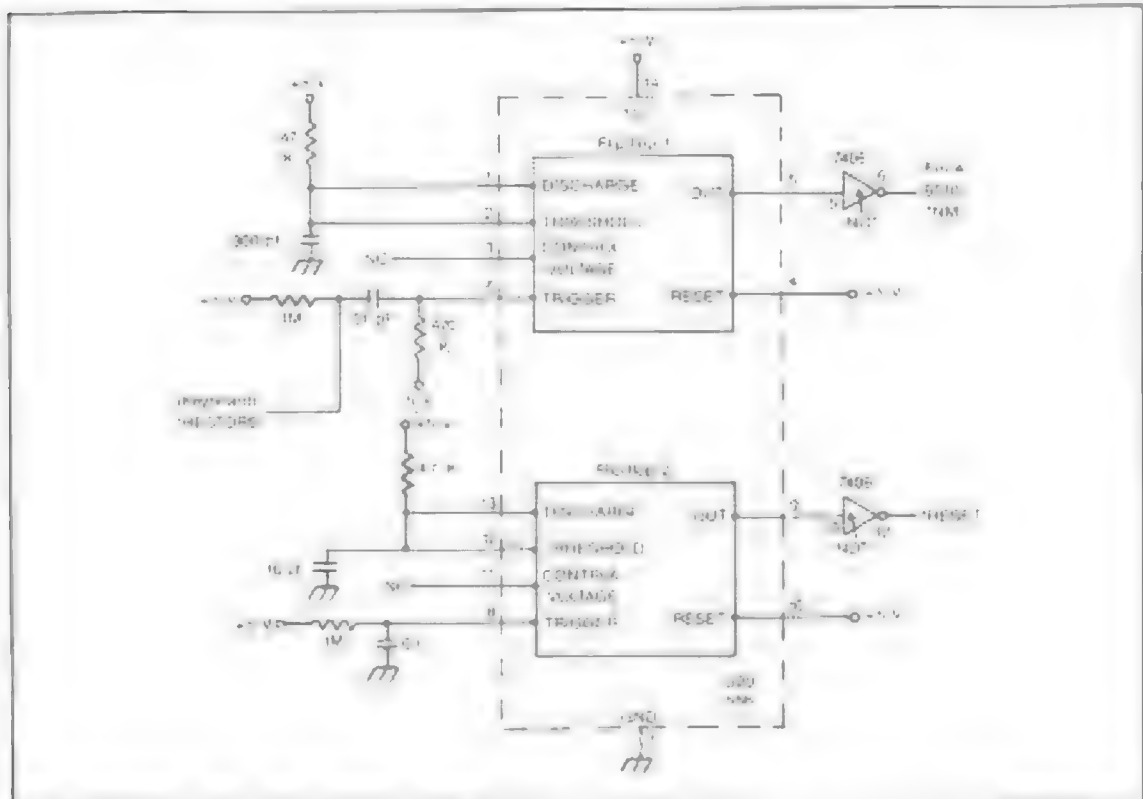


Fig. 8-27. Each 555 timer contains two separate timers. One handles the reset circuit and the other the RESTORE key.

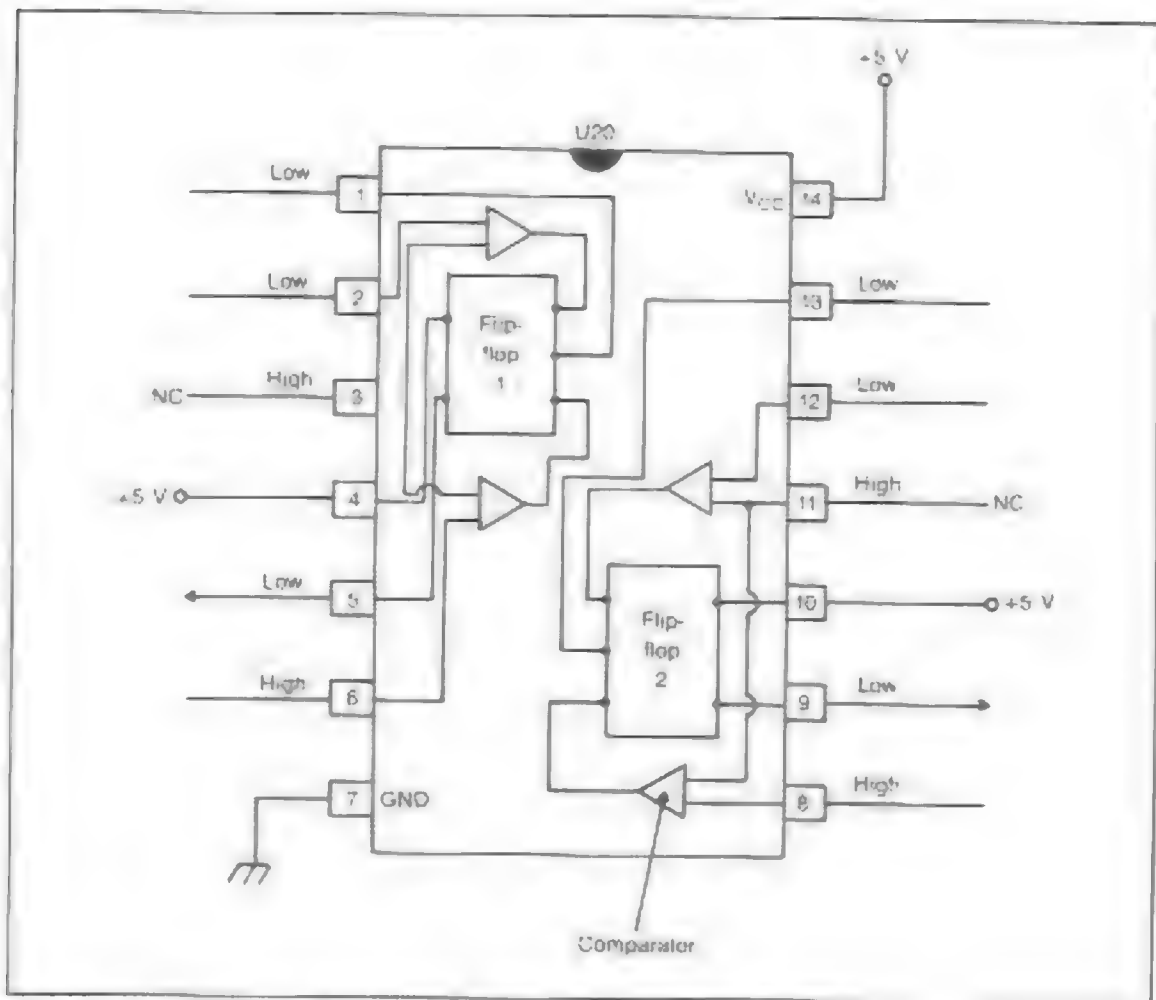


Fig. 8-28 The timers respond to the logic probe with these states.

required here in the United States area. The master oscillator in the C128 is also able to run at an alternate frequency of 17.73447 MHz. This frequency is needed to derive the frequencies required in the European area. The C128 is an international machine.

In Chapter 17 the system clocks and all their frequencies are discussed in detail, which includes U28.

U59 is an integrated circuit even though it is not contained in the usual chip. It is used as a power regulator and is part of the power supply. It is the 12 volt regulator in the supply. It outputs a smooth +12 volts dc to the C128. In Chapter 24 the power supply in its entirety is covered—this includes the external ac adapter box and the parts of the supply mounted on the printboard, of which U59 is one of the important components.

9. The System Block Diagram

Up to this point in the book, the printboard layout, as illustrated by the Chip Location Guide, has been the way to view the C128 hardware. All of the chips on the board have been examined and discussed. The discussions have been individual for the most part, showing the types of jobs that the various chips are capable of performing.

The knowledge of where the chips are located on the board, and the overview of what each chip can do, will help you to analyze symptoms of trouble and come up with chip suspects. Once you have narrowed it down to the seat of the trouble in this way, you can either try chip replacement as a repair technique or using the Test Point Charts, and taking vom or logic probe readings. If a reading is incorrect, then that is a clue.

Should an incorrect reading be found at an output pin, chances are that the chip has an internal defect. When the wrong reading is at an input pin, chances are that the chip is okay and that the circuit feeding that pin is outputting the trouble and needs a closer look. Of course, these assumptions are not

100%, but they do give you a good start on the repair. These Chip Location Guide techniques are very powerful. Used properly they will enable you to fix chip failure in at least 60% of cases.

In the book from here on, all those chips that were discussed as individuals on the printboard will now be joined together to form the C128. In order to be able to fix the remaining 40% of C128 breakdowns, you will have to know the way that this computer works. The best piece of test equipment is your brain. In order to be able to puzzle out how the machine failed, you first have to know how it works.

At this point it is time to examine the C128 hardware in detail. I'll start with the block diagram of the machine in this chapter and then examine the individual blocks of the diagram in subsequent chapters.

BLOCK DIAGRAM

From a block diagram point of view, computers today are not much different from the original ones

used shortly after World War II. The 18,000 vacuum tubes in the 1950 ENIAC have been transformed to 18,000 microscopic transistors on one chip, but the block diagram view is about the same. Figure 9-1 presents the block diagram of the C128.

The two central blocks in the diagram are based around the two MPUs: the 8502 and the Z80. Note the little auxiliary latch attached to the Z80. As mentioned, the latch and the buffer permit the Z80 to work with the data bus that is designed for the 8502.

The cores of the MPUs are the ALUs—one in each MPU. The ALU, the arithmetic logic unit, takes the input information and performs arithmetic and logic manipulations upon the data. The finished data is then output from the ALU. The ALU is surrounded inside the MPU, by numerous registers and gates. These are discussed in Chapters 12 and 13.

To the left of the MPUs are the input-only blocks. Inside the blocks are the keyboard and CIA1. The keyboard practically uses up all of the CIA1 capacity. When you strike a key, the impulse created by a keyboard row being shorted to a keyboard column is transferred to CIA1. CIA1 in turn transfers the generated pulse to a register in the 8502 or to the latch that is working with the Z80, whichever MPU is in control at the time. If it goes to the latch then the latch will input the signal to the Z80. From either MPU the keystroke is processed, stored in video RAM and appears on the TV display.

Another large group of chips in the block diagram consists of the residents of the memory map. The main job that the MPUs do is to transfer binary data bits to and from the residents of the memory map. The MPUs can be the transmitter of the data, or the receiver. When it is transmitting the data, it is writing to the map. During the receiving, the MPU is reading the data from the map.

The chips that live in map addresses are most of the large chips that we have touched on so far. The smaller support chips do not have addresses, although they are in address and data pathways. They just help out and perform specific jobs to expedite the data transferring.

In the memory map area, the first obvious chips are the 16 4164 dynamic RAM chips. The MPU works with them constantly. The MPU can trans-

mit or receive from these read/write chips. The MPU is able to store housekeeping instructions for operating the computer in some RAM locations. It is able to store the keyboard input information in a section known as video RAM. The MPU can place in RAM addresses of sprite characters. The MPU can store programs that you may write and the data that goes with the programs in RAM. The storage of binary bit data is the forte of RAM. The MPU can then retrieve any of the stored data by properly addressing the locations containing the desired bits.

The next group of memory map residents are the ROM chips. These include ROMs 1-4 and the character ROM. The addressed ROMs can transmit data to the MPU but they cannot receive data from the MPU, because its bit holders are full of factory burnt-in data. When one of the ROMs has a location addressed properly, then the MPU can read the location's contents easily.

The two CIAs are also addressed by the MPU. They are not storage places like RAM. The control registers in the CIAs have addresses. The MPU sends these registers instructions on how to operate. Once the registers are installed with the proper control bits, then the CIAs become ports of entry and exit for the C128. The MPU is then able to transmit data out to, and receive data in from, peripheral devices. The data that does travel to and from the CIAs has no storage facilities inside the C128. The data must pass through the port to wherever it is coming from or going to. The CIAs only have a few addresses in comparison to the thousands in RAM and ROM.

The PLA is connected to the MPU address bus. It only receives A15-A10 lines. It uses combinations of these bits to produce its chip select outputs such as those to the ROMs and VIC. The MMU is also connected to address lines. It uses the address bits to generate its assigned chip selects and the translated address bus, TA15-TA8 used by the multiplexing chips for RAM accessing.

VIC has 49 registers that have to be addressed with multiplexed address line inputs. This is a new VIC but the 49 registers are compatible with the older VIC in the C64. The registers are used to configure all the video outputs that the chip can produce.

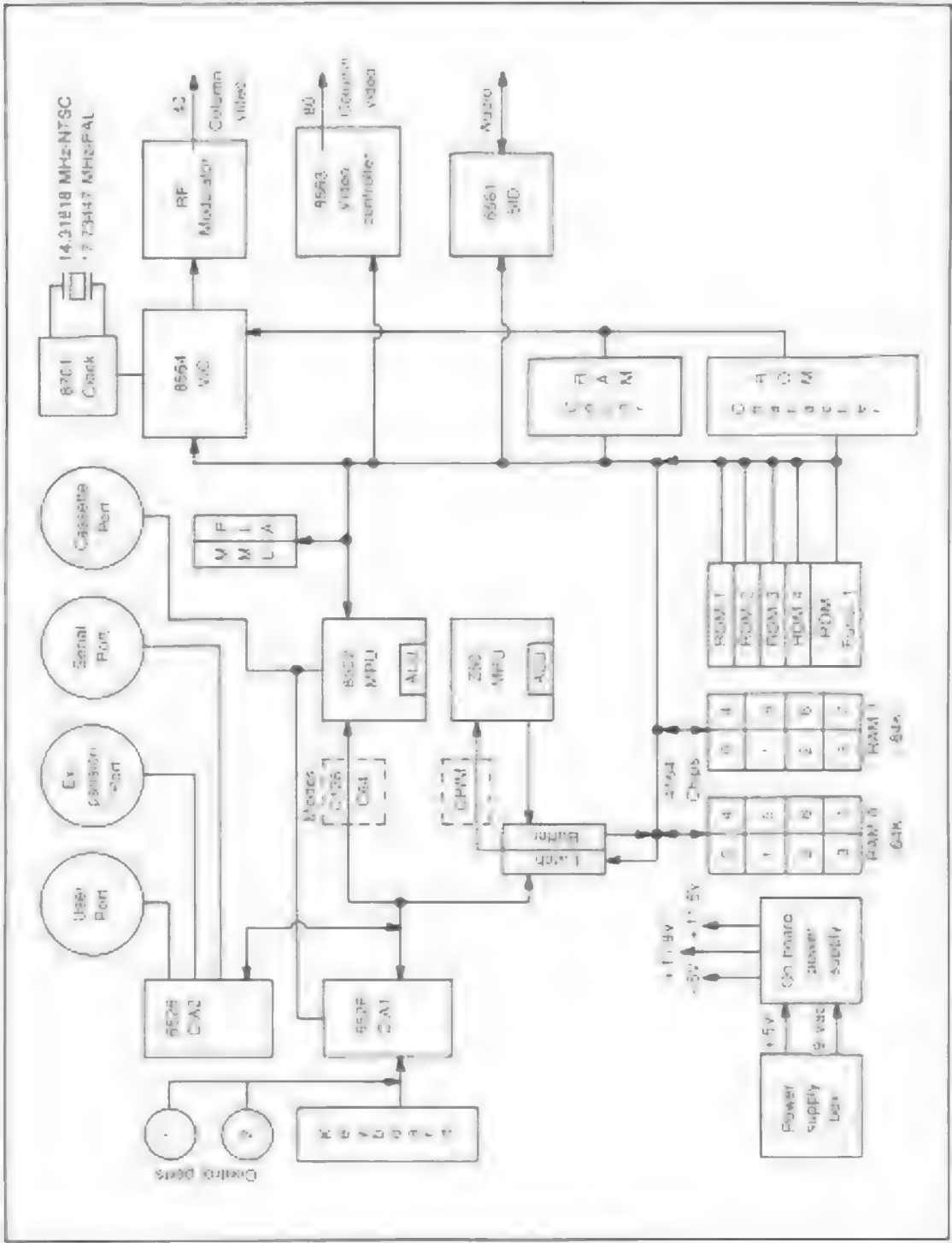


Fig 9-1 The two MPUs are the center of activity

The Video Controller also has registers that can be addressed. These registers are contacted by the MPU to produce the 80-column video outputs.

SID has its own attachments to the MPU address lines. SID operates independently of the rest of the computer except for the MPU. It receives its own inputs and produces its own outputs. It just needs the MPU, the data bus and address lines A4-A0 to operate its own registers.

The color RAM is in the memory map but has nybble locations rather than byte-sized locations. The four bits in a nybble can only hold 16 possible combinations of highs and lows. However, four bits per location is really all that is needed in the color RAM. All the color RAM is assigned to do is change colors in its respective TV screen locations. One location controls the color of one of the 1000 locations in a 40-column display. The 16 variations of bits in a location let the addressed location change the lit

block to one of 16 different colors. There will be more about this in the VIC chapter.

The overall block diagram shows that the MPU's are the originators of all addressing to the residents of the memory map. An MPU is aided in its addressing chores by the PLA and MMU. The MPU gets its operating instructions from the ROMs, uses the RAM to store data bits, gets input from the keyboard through a CIA, provides outputs to a CIA and helps the VIC, Video Controller and SID provide output. Let's examine the relationships that the 8502 has with the residents of the memory map.

THE 8502 AND DYNAMIC RAM

The 8502 processor has four types of lines connected to its 40 pins. In Fig. 9-2, you can see 16 address pins, eight data lines, seven control lines and I/O port connections. Then there are the usual +5 volt input and ground.

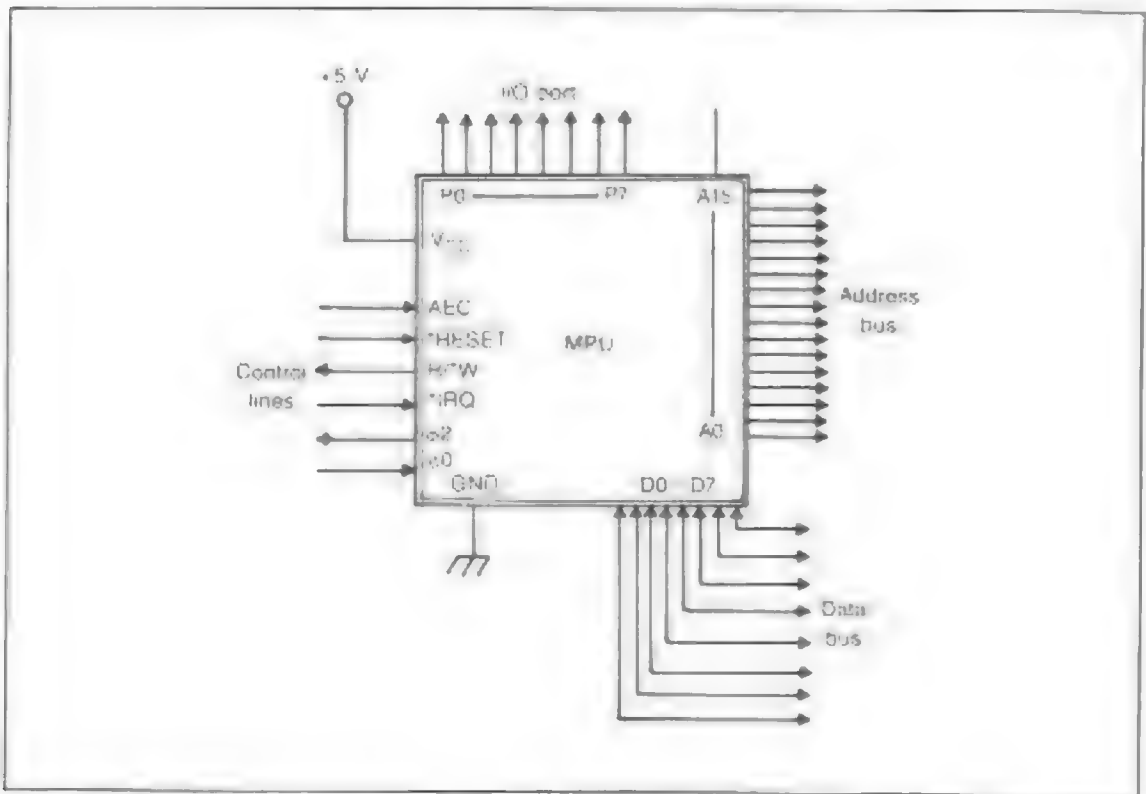


Fig. 9-2. The 8502 has four types of lines connected to it. They are the address, data, control and a one-byte I/O port.

The address lines are output only. They are the familiar A15-A0 and travel over the board to a lot of the chips. The data lines connect to the data bus, D7-D0 which also are found all over the board. The control lines are split between inputs and outputs. The I/O lines are all output types. They connect to the first two locations in RAM and do a special job that is discussed in Chapter 12.

The relationship between the 8502 and the 128K of RAM works in the following way. The data bus lines are connected directly to the 16 chips. When a location in RAM is addressed, then eight of the 16 chips are activated, the location found and data can move either into the location or leave the address. The addressing is tricky, however.

The 16 chips are broken down into two banks of eight chips each. The banks are given the numbers of 0 and 1. The first thing that the 8502 must do is to specify which bank it wants to address. The

8502 only has 16 address lines which can only address 64K of RAM at a time.

The 8502 therefore sends address bits to the MMU, as in Fig. 9-3. The MMU is able to derive signal bits from the A15-A0 input. The MMU then outputs, from its pins 12 and 11, two signals called \bullet CAS0 and \bullet CAS1. These signals are then gated in two sections of U19, the 74F32, and then sent onto pins 15 of all the DRAMs. \bullet CAS0 is connected to eight chips of Bank 0 and \bullet CAS1 is applied to eight chips in Bank 1. If \bullet CAS0 is low and \bullet CAS1 is high, then Bank 0 will be turned on. If \bullet CAS1 is low then Bank 1 will be activated.

Once the bank is chosen, then the location on the chips can be activated. The address bus from the 8502 is then sent to the two multiplexer chips, U14 and U15, a pair of 74LS257's. The connections to the chips are not straightforward, as indicated in Fig. 9-4. The 16 address lines are connected like

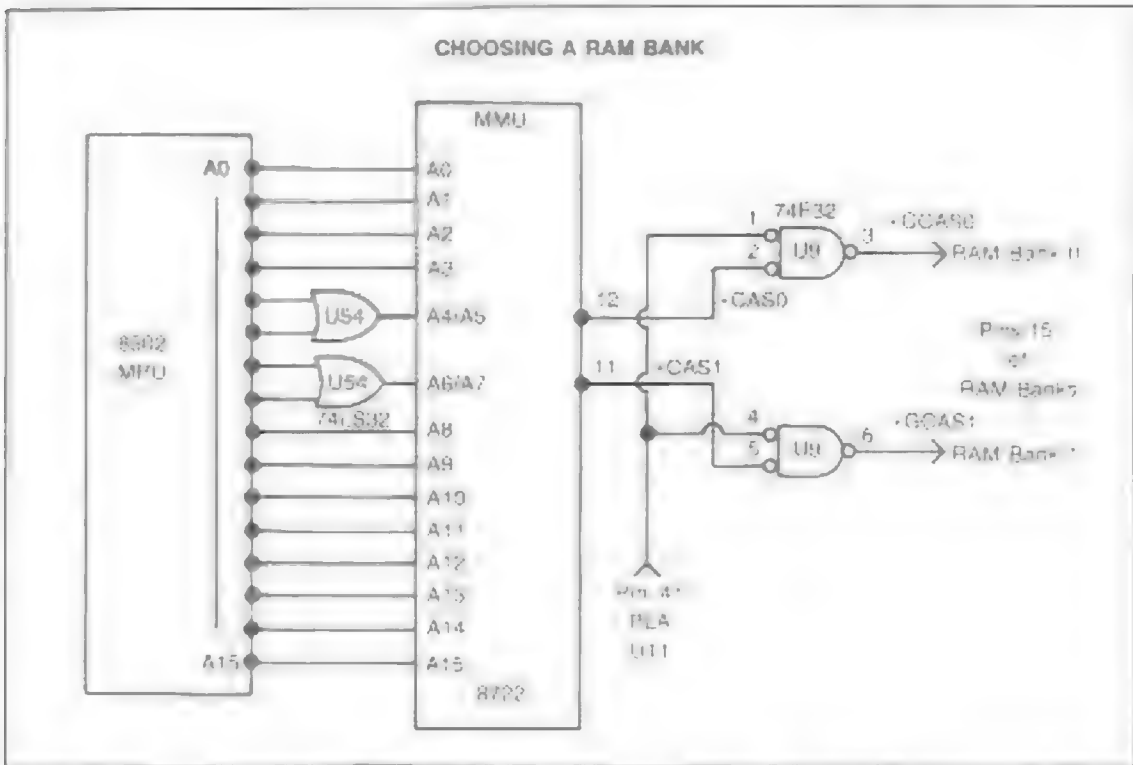


Fig. 9-3 Because the 8502 can only handle 64K of one bank at a time, the MMU and the 74F32 lend support. The 8502 generates signals \bullet CAS0 and \bullet CAS1 that choose one 64k bank over the other.

CHOOSING A RAM REGISTER

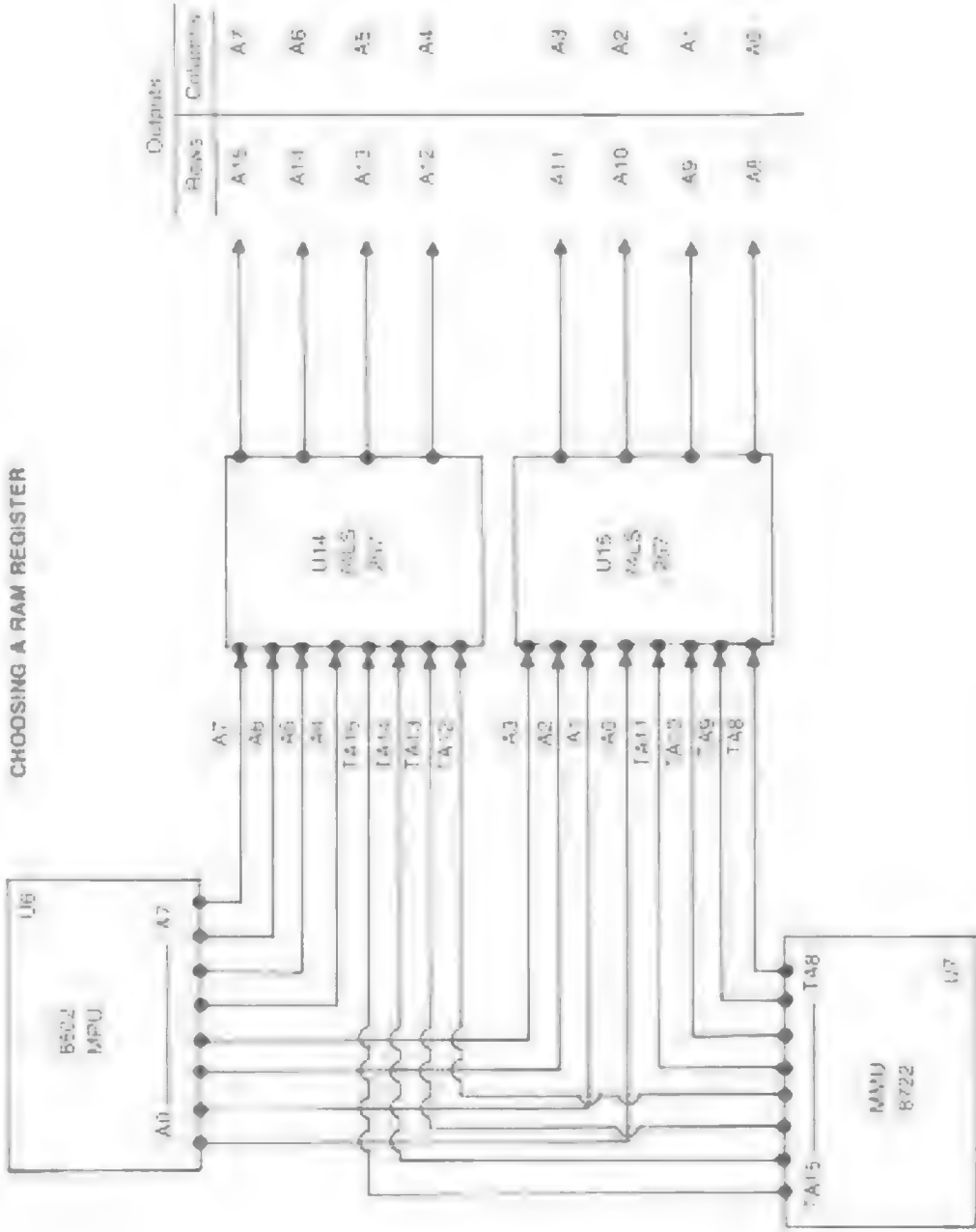


Fig. 9-4 The MPU sends eight address lines to the multiplexers while the MMU provides the remaining eight address lines

the following: U14 receives lines A7, A6, A5 and A4; in addition U14 receives from the MMU, translated address lines, TA15, TA14, TA13 and TA12; U15 receives the remaining lines and gets A3, A2, A1 and A0. It also gets TA11, TA10, TA9 and TA8.

These 16 address lines choose one of the 64K locations in the chosen bank. The multiplexers output the row address and the column address of the location. The eight bits needed to address the rows of DRAM matrixes are 15-8. The eight bits needed to address the columns are 7-0. The multiplexers with their AND and OR gate internal circuits are able to automatically output at one time the eight 15-8 address bits. A strobe signal called *RAS (row address strobe) is applied at the pins 4 on the bank of chips (See the Master Schematic.) This strobes the 15-8 bits into the row decoder inside each DRAM. The rows are thus addressed.

Next, a second strobe signal called *CAS is applied at the pins 15 of the chosen bank. This turns off *RAS and *CAS opens up the multiplexed address pins to receive the column address bits. The bits are then strobed into the column decoder and address one of the columns. The two signals *RAS and *CAS originate in VIC.

Once the desired number row and the desired number column are addressed on all eight chips, then the bits located at their eight intersections are activated. The locations are then open for business and will either receive a byte from the 8502 or send a byte copy of their contents to the 8502 over the data bus.

A direct data bus is between the DRAM banks and the 8502. If the Z80 is the MPU in control, then the data bus is not directly connected. A latch, U12, and a buffer, U13, are in the data lines to interface the Z80 with the 8502 designed data bus. The data bus lines, D7-D0 are wired one to a chip in a bank. The D7 line is wired to the D7 assigned chip that contains all 64K D7 bit locations. The D6 line is wired to the D6 assigned chip that contains all the D6 bit locations, and so on.

Each DRAM chip has two data lines, one for inputting a bit and one for outputting a bit. The two chip lines are wired together on the board, but act

separately according to whether a read or write is in progress. When a read is going on each DRAM gives up a copy of the one bit in the addressed location. During a write each DRAM receives one bit and stores it into the addressed location.

THE MPU AND ROM

128K of RAM is available in the C128. When you turn on your machine in C128 mode the sign-on message lets you know that 122,365 of the 131,072 bytes in 128K is available. Where are the missing 8707 bytes? 6144 are assigned to the Kernel and BASIC. 1024 are provided for the 40-column video, 1024 are used by the MMU, and the top 256 bytes in every bank are used by the MMU. Lastly, three bytes are needed for BASIC to mark the starts and ends of programs.

However, although there are 128K in RAM, the 16 address lines from the MPU can only deal with a 64K bank at a time. Addresses in decimal never exceed the 64K range. In order to utilize all of the available RAM, the MPU uses the help of the MMU and banking techniques.

The ROMs are also addressed in the 64K range, as seen in Fig. 9-5. This means that the ROMs have the same addresses as parts of RAM. At first glance it appears that a conflict could exist between both of them using the same addresses. Obviously, this situation cannot be tolerated or programs would crash all over the map.

What happens is the following during ROM addressing. The ROMs, U32, 33, 34, 35 and 36, if a U36 is plugged in, are all directly wired to the MPU. The address lines and the data lines all go straight to the ROMs without any other chips inbetween. When a ROM is addressed by the MPU, the 16 bits go directly. The only other addressing that is performed on the ROMs are chip selects from the PLA, to pick the one ROM out of the possible five to turn on.

ROMs cannot be written to. They can only be read. When the MPU addresses a ROM with a PLA generated chip select and the address bus bits, and asks for a read, then the CAS is disabled in both Bank 0 and Bank 1. This turns off all 128K of RAM.

ADDRESSING THE ROMs

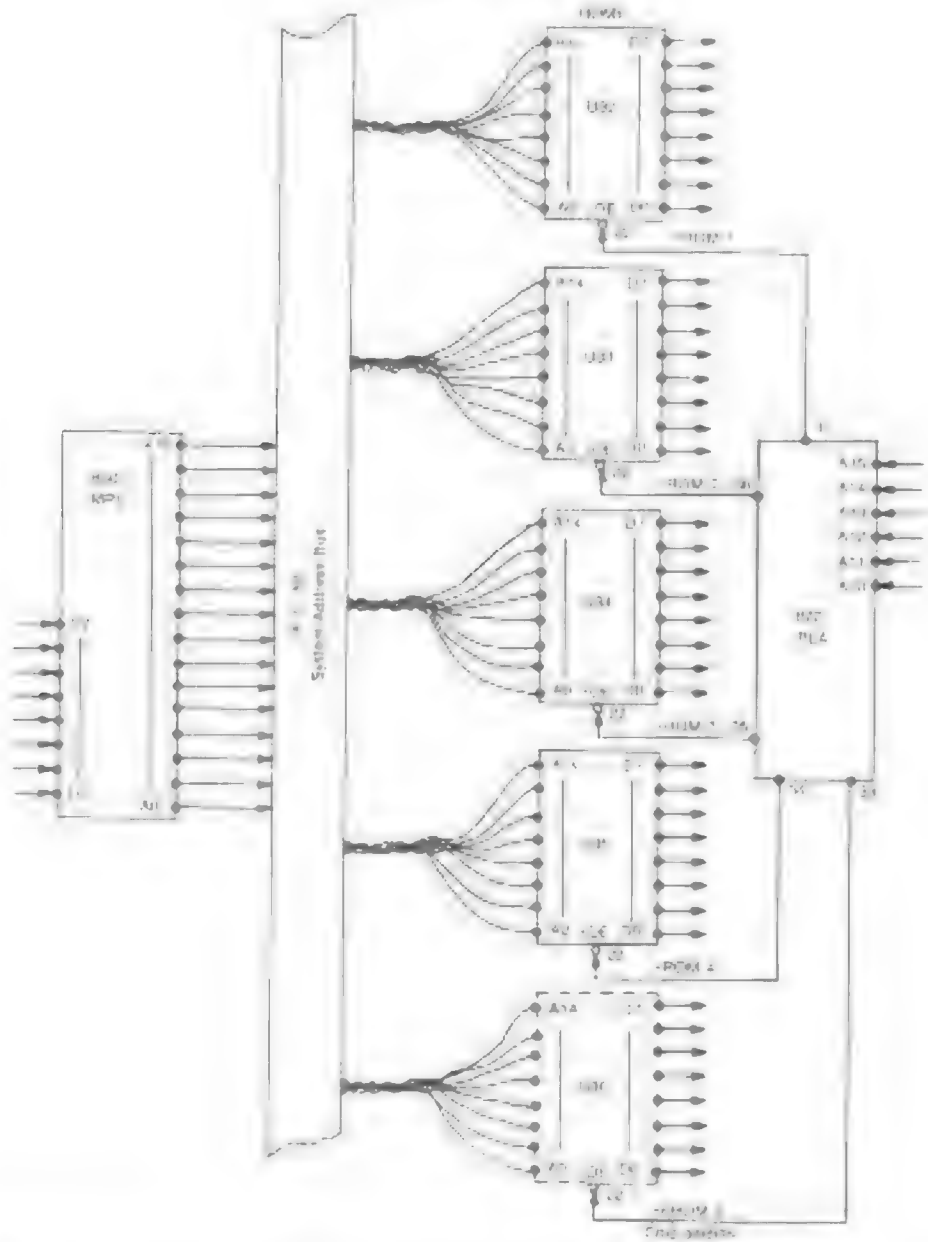


Fig. 8-5. The system ROMs are all addressed directly by the MPU because the PLA performs a chip select function. The PLA turns on the desired ROM chip and then the MPU addresses its registers.

The ROM read can then take place with obstruction. The ROM byte location addressed sends a copy of its contents to the MPU.

Should the MPU write to a ROM location, the CAS in RAM is not disabled. The ROM simply ignores the data bits that arrive at its data bus lines. The RAM is on. The data bits from the MPU, since the RAM is available, gets stored in the addressed location. Programmers can find this situation handy under some circumstances. Anyway, the ROMs are read without problems and if a write is attempted to ROM, the data ends up in RAM that has the same address.

In C128 mode, 48K of ROM is available. In C64 mode, the same amount of ROM is used as in the C64 machine. The MMU registers handle all the ROM addressing manipulation. This is covered in Chapter 15. There are many different layouts in which the ROMs can be placed.

The character ROM is in the memory map and can be accessed by the MPU. However, its main job is to be accessed by VIC and the 80-column Video Controller chip. It contains the complete character sets for both the C128 and C64 modes. These characters are designed to appear on the TV screen. When the MPU reads the character ROM in BASIC it receives binary bits that get displayed as decimal numbers.

The character ROM is actually addressed by the shared address bus, SA7-SA0, and the translated address bus, TA11-TA8. These buses are discussed in more detail in Chapter 18. Once the ROM is addressed it gives off a copy of its addressed contents to the data bus. These bits can be received by the MPU if it is the addresser. Should the VIC or Video Controller be the addresser, then they will receive the bits, convert the bits to video signals, and output them to the TV display.

MPU AND CIA INTERFACE

Each CIA only has 16 internal registers that can be addressed. Only four address lines are required from the MPU, A3-A0. Figure 9-6 shows how the address lines connect to the CIAs. CIA means *Complex Interface Adapter* and the device is a complicated miniscule machine. Chapter 19 goes into its

details. The CIA operates directly with the MPU on its internal side. On the external part it operates with peripherals.

The MPU sees the CIA as simply some more addresses in the memory map. The CIAs, though, are the ports of entry and exit giving the MPU instant communication with peripherals.

The CIAs have the usual D7-D0 data lines that connect to the data bus. The CIA can be read from or written to by the MPU, just as RAM can. The data pins connect to the inside of the computer. The CIA has 16 other data lines that connect to the peripherals. These lines are two sets of eight each so they can handle bytes of data. They are designated with the lettering PA7-PA0 and PB7-PB0. Two sets of registers, almost identical, are in the CIA. The PA lines are coming from one set and the PB lines from the other register. The D7-D0 bus lines connect to both sets. Each register set has two addresses on the memory map. The data bus will service whichever set the MPU addresses.

CIA1, U1, is used to I/O the keyboard and the control ports. CIA2 accesses the serial port, the user port and the expansion port. The two chips are identical but because of the different jobs they are performing, the Test Point Charts of each have very different readings.

Inside a CIA, the two ports, A and B, use six registers to do their job. Each register is byte-sized. In addition to these six, there are ten more registers performing other jobs. These other duties have to do with timing, interrupts, and serial I/O which is supplied besides parallel I/O. These ten registers also have their own addresses giving the registers a total of 16 address lines. The four bus lines, A3-A0 can generate 16 different sets of address bits.

The 16 addresses are opened up individually through four address pins, RS3-RS0. The RS stands for register select. RS3-RS0 are connected to A3-A0.

The CIAs have one pin called ϕ CS to select the chip. The selection bits are coming from U3, the 74LS138 1-of-8 decoder chip, as mentioned earlier. The CIAs also receive a signal called R/W (read/write line) from the MPU as shown in Fig. 9-7. The line tells them to send data to the MPU

ADDRESSING THE CIA's

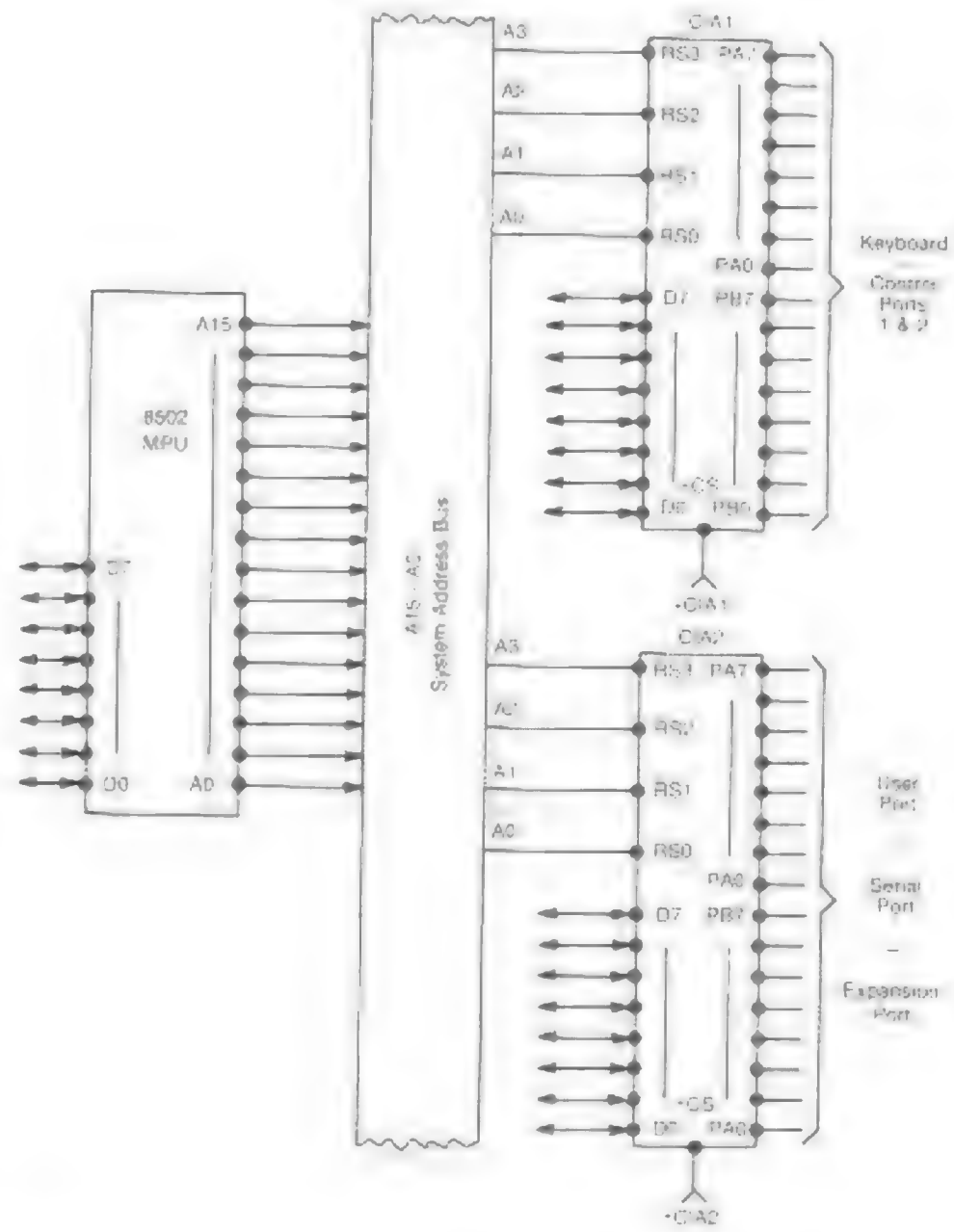


Fig. B-6 The CIAs are addressed with only four lines plus the chip select signals, $\overline{CIA1}$ and $\overline{CIA2}$.

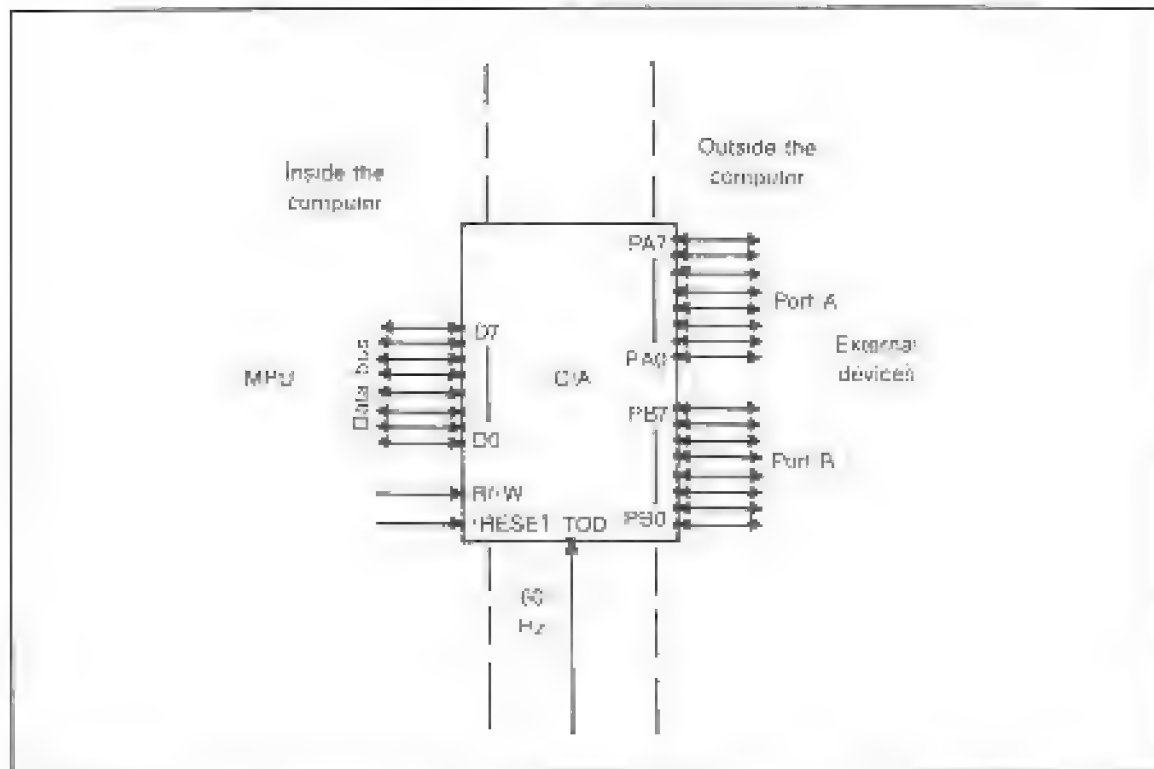


Fig. 9-7 The CIAs are equipped with an R/W (read/write) line that tells it which direction to move data. There are other control lines including *RESET and Time Of Day.

when the bit is a high or get ready to receive data from the MPU when the bit is a low.

The CIAs are initialized by a signal called *RESET when the C128 is first turned on. The CIAs are connected at pin 19, TOD, to the electric company's accurate 60 Hz frequency. This makes it run like an electric clock through this Time Of Day connection. Internally, four of the CIA's registers are used to keep track of AM/PM hours, minutes, seconds and 10ths of seconds. More detail on the CIAs is in Chapter 19.

THE MPU AND VIC

The MPU can address the VIC registers. VIC has 49 registers. The older 40-pin VIC found in the C64 only had 47 registers. This newer 48-pin VIC duplicates the 47 registers the old version had, plus two more. One of the additional registers allows extended keyboard scanning. Three additional key-

board control lines are scanned with this register. This gives the C128 keyboard additional keys in the C128 mode while still remaining compatible with the C64 mode.

The other extra register lets the VIC generate a new faster 2 MHz clock. These registers are discussed in more detail in Chapter 20.

In addition to the MPU addressing VIC, VIC has a 16K addressing ability of its own. It can access a 16K group of locations in the memory map. During its operation, VIC has to be able to access parts of RAM where video data is stored—it must constantly use the character ROM and the color RAM.

VIC's pins are an assortment of inputs, outputs and input/outputs. As you can see in Fig. 9-8, the inputs include the chip select, *CS, the color clock input drive, Phase In, the R/W line and the address lines to choose one of the 49 registers. VIC is the generator of, and as a result, outputs signals such

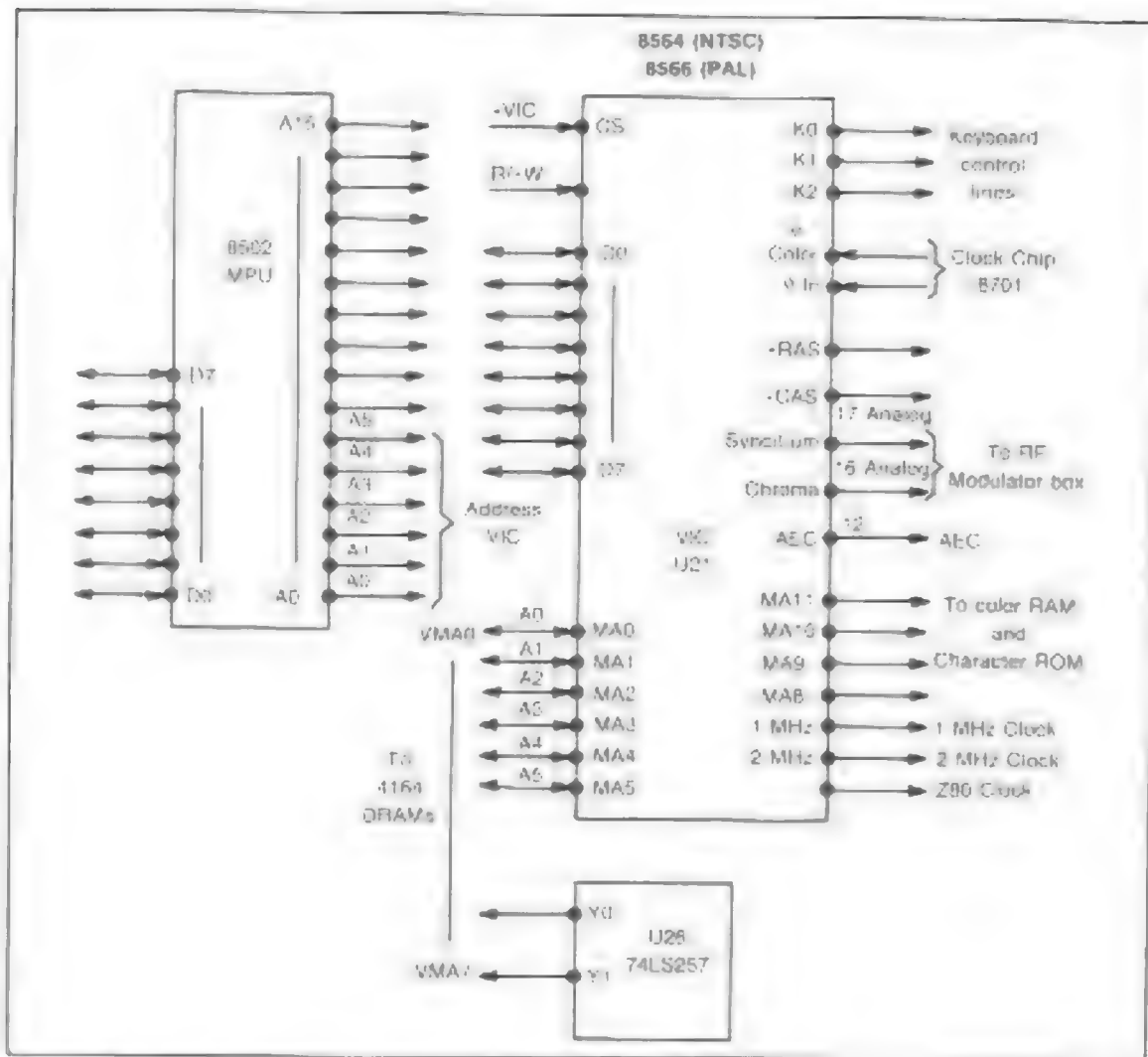


Fig. 9-8 VIC is a complex chip that can take over the computer and act like an MPU.

as \bullet RAS, \bullet CAS, Sync/Luminance, Chroma, the 1 MHz and 2 MHz clock signals, the Z80 Phase 1 clock, the color Phase 1 clock and AEC the address enable control. Among other lines are the usual D7-D0 bus connections and address lines leaving VIC to access a chosen 16K portion of memory.

In Chapter 20 the details are covered, but for now, it should be noted that some of the VIC's address lines are two way streets. The processor uses them to access VIC registers and VIC uses them to access video RAM, character ROM and color

RAM. When the processor is accessing VIC the lines to VIC are inputs. When VIC is in control the lines become outputs.

Out of pin 12 VIC sends out the AEC (address enable control) signal. AEC is an important control. When AEC is a high the MPU is in charge of the circuits. When AEC goes low, the MPU address lines are disabled and VIC takes control.

When VIC is in charge it reads the video RAM section 60 times a second. The video RAM is the storage area where the codes for the characters that

are displayed on the screen are kept. VIC keeps scanning video RAM and with the help of the character ROM continually updates the characters being displayed.

The inputs to VIC are all digital signals. The outputs just mentioned are also digital signals. Besides these inputs and outputs, the VIC also outputs three analog signals. Inside VIC the digital inputs are converted to analog. These analog signals are combined to form color TV video. This video output is quite like the ones you receive from a commercial TV station.

Pin 17 outputs the sync and luminance parts of the VIC formed signal. They are fed into the RF Modulator box. Pin 16 outputs the color signal. It also goes into the RF Modulator box.

RF Modulator

Inside the box the signals encounter a color mixing and amplifying circuit, shown in Fig. 9-9. The luminance signal is injected into the base of a 2SC458 npn transistor. The color is passed through a coil and capacitor and is injected into a second 2SC458 transistor. The signals are processed in the transistors and are then passed out of the box, from box pins 6 and 7, to the composite video output port to pins 1 and 6 of the port, as in Fig. 9-10.

Besides their individual exits the luminance and color signals are mixed together in a small RC circuit. Their resultant, a composite color TV signal, is then passed out of the box to pin 4 of the composite video output port. Note that this composite color TV signal is separate and is used independently of the luminance and color outputs.

In addition to these two separate TV display signals exiting the box, a third TV display signal is generated. A 2SC460 transistor is in the box. It is in an oscillator circuit that is running at a TV frequency. A channel select switch can set the oscillator to run at either Channel 3 or Channel 4. The switch is set on 3 in Fig. 9-9.

The 2SC460 circuit is outputting a Channel 3 carrier wave. The carrier emerges from the twin IN4148 diodes and connects to a line. The line contains the composite color TV signal. Incidentally, the

line also contains the audio signal that SID is sending out through the audio circuit from 2SC460 npn transistor.

The Channel 3 signal then mixes with the composite color TV signal generated by VIC and the audio signal produced by SID. These signals are cleaned up through an RCL network at the end of the line and output through the RF output port. This signal can be connected to any commercial TV, and viewed and heard on Channel 3.

VIC is a lot of things, but mainly it is the I/O chip for the color video. The circuitry input to VIC is tested and probed with the vom and logic probe. The digital output from VIC that goes back into the digital circuits also can be examined with the vom and logic probe. There are no real reasons, during normal servicing, in the digital circuits, to use a scope. You are only dealing with highs, lows and digital pulses.

Once the analog color TV signals leave VIC, the servicing techniques are changed. The analog signals are forms of TV impulses. They can be checked accurately with an ordinary TV scope. The waveforms the scope should display are found in Chapter 20.

THE MPU AND SID

The 6581 Sound Interface Device is the music synthesizer and sound effects system to go with the arcade style graphics the C128 is capable of producing. SID is a 28-pin DIP. The number 6581 is compatible with the 8502 since the 8502 is simply an improved 6502.

There are 29 addressable registers in SID. Five address lines are able to contact 32 locations. SID has five address lines A4-A0, as in Fig. 9-11. The address lines can set up a read or write from the MPU to these locations. Should a read or write be tried to one of the three remaining addresses in SID that do not exist, a read returns garbage and a write is simply ignored. The SID, from the MPU's viewpoint, is considered to have 29 legible memory addresses.

In Fig. 9-11, there are the usual eight data pins, D7-D0 at 22-15. The MPU supplies data to SID dur-

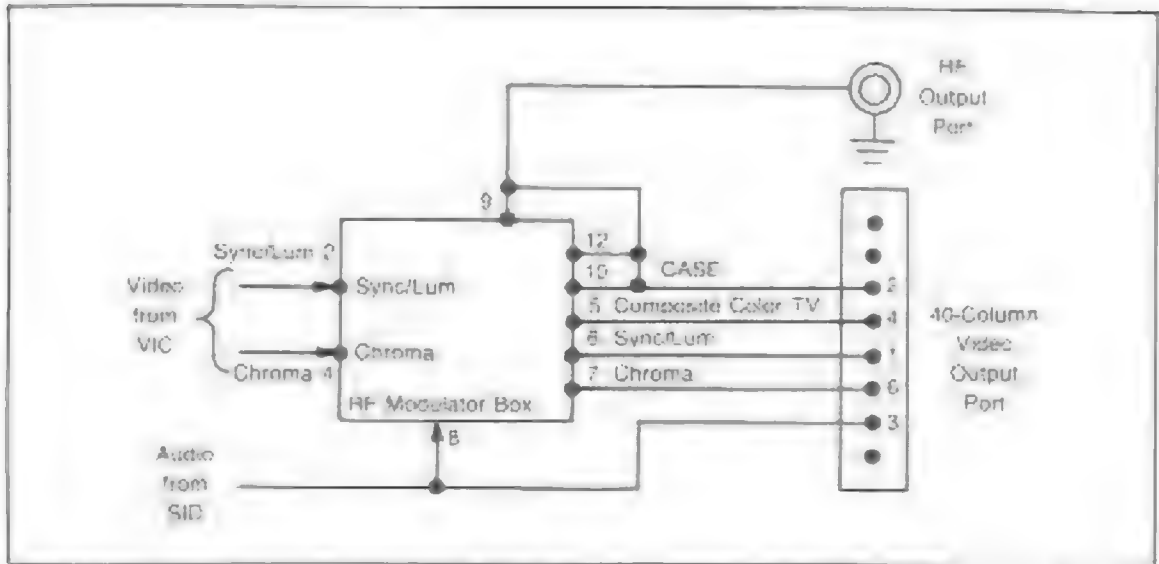


Fig. 9-10. Three separate outputs come from the RF Modulator box, all in the 40-column mode. One is the RF Output for a home TV. The second, from pin 4 of the video port is a composite color TV signal that works in a monitor. The third is a combination sync/lum-chroma signal, from pins 1 and 6, that will work in a special direct monitor.

it is routed to the SID pins. The signals set the position of potentiometers of the chip. The Audio In and the CAPs mix and filter the audio. This is covered in the SID chapter.

SID, after all the waveform generating and modulating has one audio output at pin 27. The output can have a peak-to-peak maximum of two volts. There is a dc level of six volts and this can be coupled to any audio amplifier. On the printboard, Q2 is an audio amplifier transistor that receives the SID audio output and sends it on. Q2 is found to the left and up near the top of the SID DIP, on the printboard.

Q2 sends the audio output signal to both pin 3 of the composite video port plug and to pin 8 of the RF Modulator box, shown in Fig. 9-10. The audio can then be taken from the port plug and is also modulated into the TV Channel 3 or 4 frequency as sound to go with the video.

SID is a complex chip that contains both MOS and TTL components. The Master Schematic shows V_{DD} , pin 28. The chip gets +12 volts to power the MOS parts. At pin 25, V_{CC} , the chip receives +5 volts to energize the bipolar transistors. SID has a

•RESET line and a clock input to keep it in time with the pacing of the MPU.

THE MPU AND THE VIDEO CONTROLLER

The MPU sees only two registers on the 8563, starting at decimal 54784. When that address is contacted, it is through U3, the 74LS138 1-of-8 decoder. The decoder emits a signal called •CS8563. This turns on the chip at pin 7, •CS. On pin 8, •RS, address line A0 enters and selects one of two registers with either a high or a low.

Actually there are 37 registers in the 8563 but they must be contacted indirectly through the original two, an address register and a data register. The details of the register addressing is described in Chapter 21.

Besides being able to be accessed by the MPU, the 8563 can do accessing on its own like the VIC can. The 8563 can do things as the VIC does. It does not duplicate VIC, since VIC is used for all the 40-column display and the 8563 performs all the 80-column video output chores.

VIC uses a section of RAM for storing its display codes that are appearing on the TV screen. As

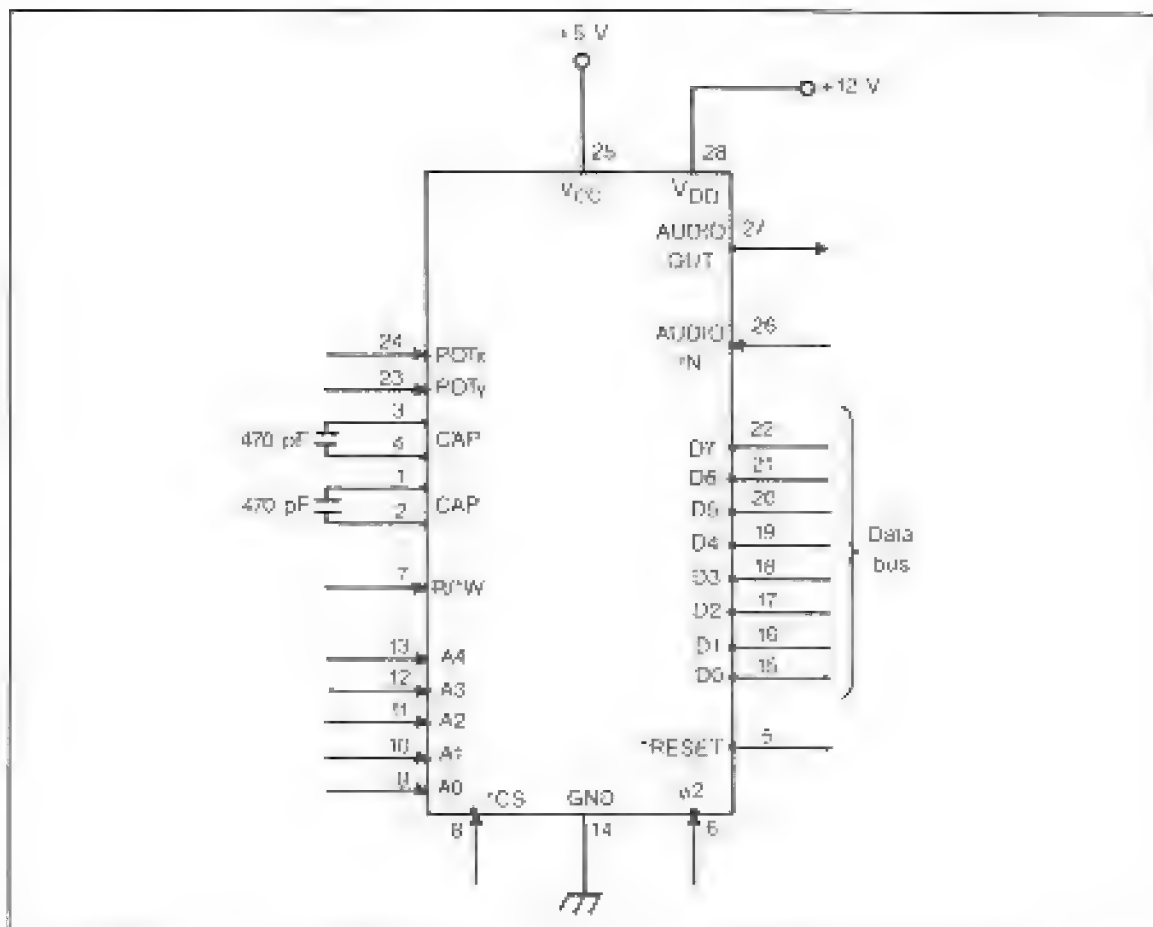


Fig. 9-11. SID communicates with the MPU through the eight-bit data bus. SID's registers are eight-bits wide.

mentioned, VIC scans that section of RAM, dubbed video RAM, so it can constantly update whatever is on the TV screen.

The 6583 Video Controller does a similar job when it is displaying its 80 columns. However, the 6583 has its own personal RAM. There are two RAM chips, U23 and U25, a pair of 4416 chips that the 8563 works with. Note the three different sets of data pins on the 48-pin chip in Fig. 9-12. DA7-DA0 and DD7-DD0 operate with the two 4416 chips. D7-D0 connect to the system data bus. The two 4416's provide the 8563 with 16K of personal video RAM. The 8563 also generates the refresh signal

for the two 4416 chips. It also provides the RAS, CAS, write enable (R/W), address and data signals for its personal RAM.

The video output signals of the 6583 are R, G, B and I. The 80 columns are most useful with text displays. The RGBI output produces the clearest text display because it provides the best form of picture resolution available in the C128. The 8563 outputs to U24, the 74LS244, an octal driver. The driver or buffer, amplifies the various signals and sends them to the RGBI output port. Chapter 21 covers the 8563 in greater detail.

10. Servicing the Logic Gates

In order to intelligently troubleshoot a computer, you should have an understanding of what is actually happening. A computer works by transferring highs and lows, positive 5 volts and 0 volts. Actually the voltages do not have to be exact. They can vary somewhat, but that is not important.

The highs and lows are electrical code for 1 and 0. The computer then works by generating and transferring 1's and 0's. The 1's and 0's, which you know as bits, are transferred through gates and registers. Gates simply process and pass bits, but registers can store them. All there are in digital circuits are gates and registers handling bits. When a gate or register circuit fails, it doesn't handle bits as it was designed to do--and trouble occurs. Troubleshooting consists of seeking out the defunct circuit. The repair is complete when the circuit is replaced with a good one, or is somehow fixed and works once again.

When the 8502, Z80, VIC or Video Controller chip addresses a location, it sends out binary bits. In most locations, eight binary bits are being stored.

In some locations, like color RAM, only four binary bits are stored. The 16 address bits can form 65,536 different combinations. Each combination represents one number between 0 and 65,535. Zero is a full fledged number and there is a location called 0. The eight bits in a byte-sized location can form 256 different combinations representing numbers between 0 and 255. In the color RAM nybble locations there are four bits. They can form 16 different combinations representing numbers 0 to 15.

When you write a BASIC program, or run a test with BASIC, you must code the binary combinations into a decimal number. Should you be writing in machine language, the decimal number is not recognized. Machine language needs a different numbering system to be recognized, called hexadecimal or simply *hex*.

For the most part, you can run tests in easy BASIC. However, it is a good idea to be familiar with the relationship between binary bits, decimal code for the binary bits and hex code for the binary bits.

The rest of this chapter is a briefing session on

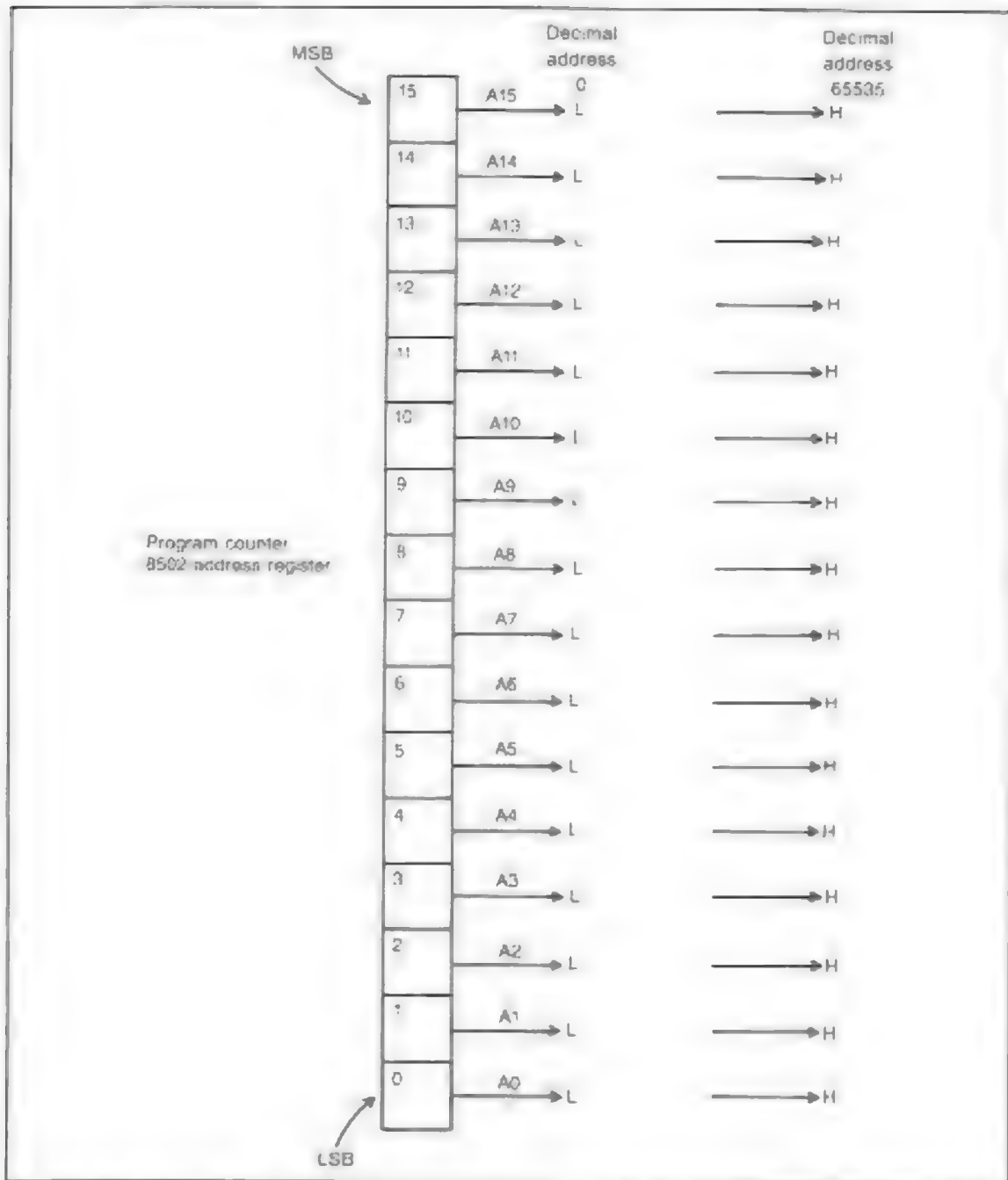


Fig 10-1. The first possible set of bits in the PC codes decimal address 0. The last set of bits codes decimal 65535

binary bits and how they are handled and coded in gates. The next chapter is about binary bits and how they are processed and stored in registers. Between the two chapters, you will take a giant step towards being able to understand what is happening in the deepest circuits of the C128.

DECIMAL AND BINARY

Decimal numbers are used in the Commodore 128 BASIC to list addresses. In the 8502, there is a 16-bit register called the program counter. It does most of the 8502's addressing chores. You have seen where a row of 16 highs and lows can form 64K combinations. Each combination of highs and lows in the program counter can form one of the 64K addresses on the machine's memory map. The program counter bits are connected to A15-A0 of the address bus.

The first combination of bits in the Program Counter is LLLL LLLL LLLL LLLL. The row of lows in Fig. 10-1 is called address number 0 in decimal. (There are no spaces in the actual register. I put them there for easier reading of the 16 bits.) The program counter automatically places these bits onto the address bus. As soon as these bits leave the MPU, the program counter is incremented by one automatically. The PC is built to do this.

The next address is LLLL LLLL LLLL LLLH. This group of bits will address location 1 in decimal. The next address is LLLL LLLL LLLL LLHL. This is address number 2. The program counter continues its incrementing. If it is not stopped, it will mindlessly keep counting up to HHHH HHHH HHHH HHHH. This is location 65,535. It is the 65,536th physical location, since the first location has an address of 0.

Let's add up the Hs and Ls and see how they relate to the decimal numbers, or in other words, how can they be coded back and forth. It really is easy to crack the code. I'll do it with 16 bits, and then you can handle any size register. Other registers you might want to code either way could be the 8-bit ROM locations or the 4-bit Color RAM

The bits in the address bus are names A15 through A0. A15 is called the MSB for the Most Significant Bit. A0 is called the LSB for the Least Significant Bit. The bits in between are either higher or lower according to what bits they are referenced from. The LSB A0 when related to decimal has a value of 0 or 1 according to whether it is storing a low or a high. This, of course, seems obvious. Not quite as obvious is the next higher bit A1. It has a decimal value of either 0 or 2. A low is 0 and a high is 2. The next higher bit A2 has a value of 0 or 4. The values continue to double on a high till bit A15 has a value of 0 or 32,768.

In order to convert a register to its decimal code, all you have to do is add up the values of all the bits. Any bit holder containing a low is counted as a 0. All bit holders storing a high is counted by its significant value. The place values are given in Table 10-1.

With the aid of Table 10-1, you can convert highs and lows to addresses and vice versa. For instance, suppose you are probing the address bus and you find a fixed set of bits on the bus. The bus is stuck on that number and will not respond to any attempt to change it. You want to know what address the bus is pointing to. The bits are HLLH

Table 10-1. Binary to Decimal Conversion.

Significance	Low	High
LSB A0	0	1
A1	0	2
A2	0	4
A3	0	8
A4	0	16
A5	0	32
A6	0	64
A7	0	128
A8	0	256
A9	0	512
A10	0	1024
A11	0	2048
A12	0	4096
A13	0	8192
A14	0	16384
MSB A15	0	32768

HHHH LLLL LHLL. The MSB is the first H and the LSB is the last L. The address can be converted to decimal by adding the values of the bits together. The conversion would look like this:

Binary to Decimal		
MSB	H	32768
	L	0
	L	0
	H	4096
	H	2048
	H	1024
	H	512
	H	256
	L	0
	L	0
	L	0
	L	0
	L	0
	H	4
	L	0
LSB:	L	<u>0</u>
Address:		40708

The address on the stack bus is decimal 40708. On the 64 memory map, 40708 is an address that is used by a cartridge ROM. This bit of information is a clue to indicate where and what is causing the stack bus.

Sometimes it is useful to code a decimal number to its binary equivalent. There could be a case where the 128 will respond to certain addresses but not to others. One of the address lines could be shorted to ground or to another line. Suppose one of the addresses that will not work is 1024. If you convert decimal 1024 into binary, you might get some idea of what line is inoperative.

To convert the decimal, you look for a combination of bits that will add up to 1024. That one is easy. A10 is exactly 1024. Its address is LLLL

LLHL-LLLL LLLL. It looks like address bus line A10 is out of commission. It can be tested for shorts or an open.

That was an easy conversion. Suppose you had to code 53281 to binary. That would take a bit of number juggling. The first thing to do is locate the closest bit value that is less than the desired number. Then keep adding values till you arrive at the correct binary code for the decimal. 53281 is converted in the following manner:

Decimal to Binary		
MSB	32768	H
	16384	H
	0	L
	4096	H
	0	L
	0	L
	0	L
	0	L
	0	L
	0	L
	0	L
LSB:	<u>1</u>	H
Total	<u>53281</u>	

The conversion back and forth is not difficult. Coding binary into decimal is simply a matter of adding the significant decimal values of each bit together. Getting the binary bit layout from a decimal number requires breaking down the number into a group of decimal numbers that consist of decimal significant values. Then the corresponding bits are arranged from MSB to LSB. Number conversion is a must during some repair work with the C128.

HEXADECIMAL

Most of the time during troubleshooting and repair work on the C128, decimal and binary coding will suffice. The BASIC ROM comes on using decimal. The logic probe and vom reveal binary highs and lows on the various test points. If you can intelligently relate the two representations of voltages, you can do practically all the jobs where numbering comes into play.

There are some occasions though, where it would be handy to use hexadecimal too. Hex is just a third way to express the decimal or binary values. It is used extensively in machine language programming. Its use during repair jobs is to code binary in another way.

Hexadecimal means six plus ten. That is, it is a numbering system where you count ten numbers from 0 to 9 and then count on up to 15 with A, B, C, D, E and F. After F comes 10. Hex lends itself to computers because a nybble of binary numbers counts from 0 (LLLL) to 15 (HHHH). The hex numbers line up with binary exactly. Table 10-2 shows the binary equivalents for hex numbers 0 through F.

Hex is a convenient way for programmers to use binary. Table 10-2 shows that one hex number can represent four binary voltage states. That is why I put a space between every four voltage levels.

Table 10-2. Binary Equivalents of Hex Numbers.

Hex	Binary
0	LLLL
1	LL LH
2	LL LL
3	LL HH
4	LH LL
5	LH LH
6	LH HL
7	LH HH
8	HL LL
9	HL LH
A	HL HL
B	HL HH
C	HH LL
D	HH LH
E	HH HL
F	HH HH

Each space is between one hex number. The four voltage levels are one hex number. It is much easier for a programmer to code one hex number than four voltage states, four 1s and 0s, four trues and falses, four sets and resets, or whatever way some engineer or programmer decides to call the logic states.

To convert any binary number to hex, put a space between every group of four bits. Then find the hex equivalent for each group. The total bit size of the binary number is not important. As Fig. 10-2 shows, there are four bit holders in each location in the Color RAM. You can express the bits held in each location with one hex number. There are eight bits in each ROM location. These location contents can be written in two hex numbers. The addresses in the C128 are 16 binary bits each. All the addresses in the C128 can be described with four hex numbers. Refer to Fig. 10-2.

PEEK AND POKE

There are two routines in the BASIC ROM that you'll find are very useful during repairs. The routines can be used normally when the C128 has trouble but it is signing on with the READY and the blinking cursor. If the trouble is preventing the sign on message from being displayed, then the following service measures can't be used.

Assuming the C128 does sign on and there is trouble, you can signal trace the residents of the memory map with the BASIC commands PEEK and POKE. PEEK lets you read the contents of a location. POKE gives you the power to stick a byte into any location that normally accepts bytes. These two capabilities are excellent troubleshooting methods.

PEEK is a function. It is not a command. It needs a command to work. The usual command is PRINT. That way you can see on the screen what was in the location you peeked into.

To get a fast look at location 1024, you type PRINT PEEK (1024) and hit the Return key. Whatever bits are in 1024 will appear on the TV screen. However, the bits will be coded in decimal.

Location 1024 is the first place on the TV screen at the upper left hand corner. If the character P is being displayed, the code 16 will respond to the

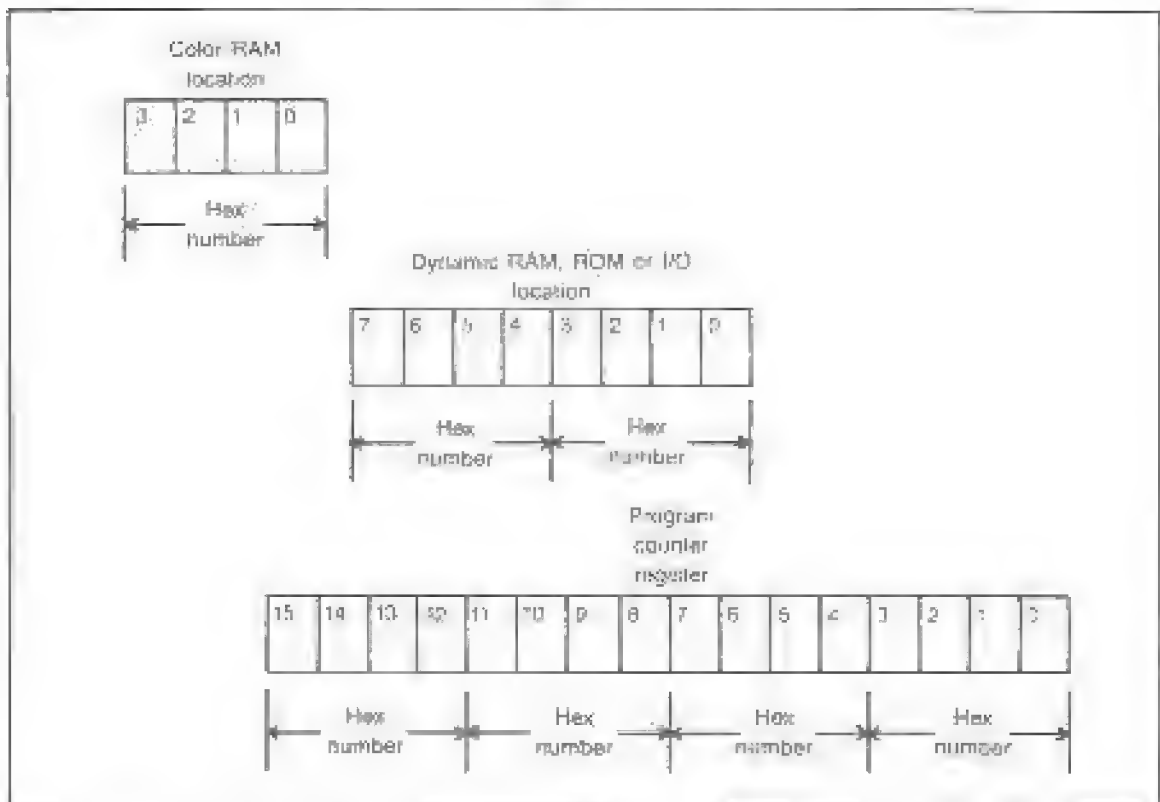


Fig. 10-2 A hex number is used to code four binary numbers. Two hex numbers can code a byte and four hex numbers the 16-bit PC.

PEEK: This is the code the VIC picks up every 1/60th of a second to update the screen. Code 16 tells the VIC to display a P.

POKE is a command. If you want to **POKE** the number 7 into location 53280, you type **POKE 53280, 7** into the machine. The BASIC routine runs the number 7 through the 8502 and onto the data bus. Meanwhile the MPU outputs the binary states for decimal 53280 and opens up that location. The binary state for decimal 7, LHHH, is installed into the four bit holders of the location. The **POKE** is complete.

Location 53280 just happens to be the place where the border color around the TV screen is stored. The LHHH forces the border to go yellow. This is a test. If the border follows instructions and goes yellow, then all the components that carried

the command, from the keyboard to the TV screen are okay. If the border does not go yellow, this is a symptom and could lead to pinpointing the source of a trouble.

The **PEEKs** and **POKEs** are powerful servicing tools. They can signal trace a great deal of the circuitry in the C128. Of course you must understand what should be happening in the circuits when you try to **PRINT** a **PEEK** or force a **POKE** into a location. There will be more about **PEEKs** and **POKEs** throughout the rest of the book.

GATES

Most of the circuits in the C128 are comprised of gates and registers. Registers are covered in detail in the next chapter. The logic states that travel through the digital parts of the computer are transpu-

lated by the gates. Most of the manipulation consists of either maintaining the voltage in a bit or changing the state of the voltage. The states flash around the circuits in times that are measured in nanoseconds (billionths of a second) and are changed from +five volts to 0 volts and back. Refer to Fig. 10-3.

The registers are storage areas for the bits. You can place a high or a low into a register bit and the register will hold the bit for as long as the correct power is applied. The registers are found in various sizes. There are *nybble registers* that hold four bits, *byte size registers* holding eight bits and *word registers* that are 16 bits across. The registers hold the bits through three general methods. In ROM chips, the bits are burnt in permanently. The latches and static RAM chips use flip-flops. Dynamic RAM stores voltage states in the capacitance formed in the insulated gate and ground of the MOS chip's insulated gates in the FETs.

With careful logic state changing in the gates and the reliable storage of the registers, computing takes place. When trouble strikes, it is often due to failure in a gate or register. The electronic circuits on a chip can and do short, open, or spring a leak under voltage or temperature pressure. Figure 10-4 shows some of the breakdowns that can befall a chip.

To troubleshoot these types of failures, it is essential that you understand what is happening to the voltage states as they make their way through the gates and the registers. That way your vom and logic probe testing will make sense and point you to a trouble quickly.

There are a lot of gates in the Commodore 128. Most prominent are the 14 pin DIPs the 7406 and the 74LS08. The 7406 has six inverters and the 74LS08 contains four AND gates.

Not as well seen are the gates in the 74LS257. Each of these 16-pin DIPs have three inverters, eight AND gates and four OR gates. While the 74LS08 has its gates separated with a pin assigned to every input and output, the multiplex chips have the gates internally wired together. The AND outputs are wired to the OR inputs. You can't put a probe on the AND-OR connections.

The 74LS238 decoder also contains internal gates. The Character ROM has an AND gate in its chip select circuit. All of the large chips have various gates. The peripheral devices contain gates. Any and all of the digital circuits have dependence on gates. Gates, like any other electronic circuit, fail occasionally. You must know how to handle gates if you want to be able to repair your Commodore 128.

The gates of concern for the C128 are the YES, the NOT, AND, OR and NAND.

YES Gate

As the name implies, a gate is a barrier in a pathway that can open to let signal pass or close and block passage. The YES gate performs that task exactly. It has two connections, one input and one output. If a high or low arrives at the input and is permitted to pass, the same type of signal will leave at the output. That is, if a high arrives, a high will leave. Should a low arrive, then a low will leave. See Fig. 10-5. There is no change made to the logic state. You might ask, if there is no change in state, why bother?

The YES gate is an amplifier stage. It is needed to amplify current levels and to match impedances from one circuit to another. It gets its YES name because the signal does not make any logic state changes as it passes from one stage to another. YES gates are found in the large chips of the C128. They are the buffers that are mentioned on occasion.

Inside a YES gate is a transistorized amplifier. When the chip is using bipolar components the amplifier could be based around an npn. The circuit is simple, as Fig. 10-6 shows. The npn is powered normally through the +5 volt supply. The high or low is input at the emitter. The output is at the collector. When a signal is input at an npn emitter, there is no phase reversal. The signal is amplified and the same logic state that was input is output. The only changes are the power output is increased and the impedance of the output is designed to match the next circuit.

The actual internal wiring of a YES gate is more extensive than this example circuit, but that is the

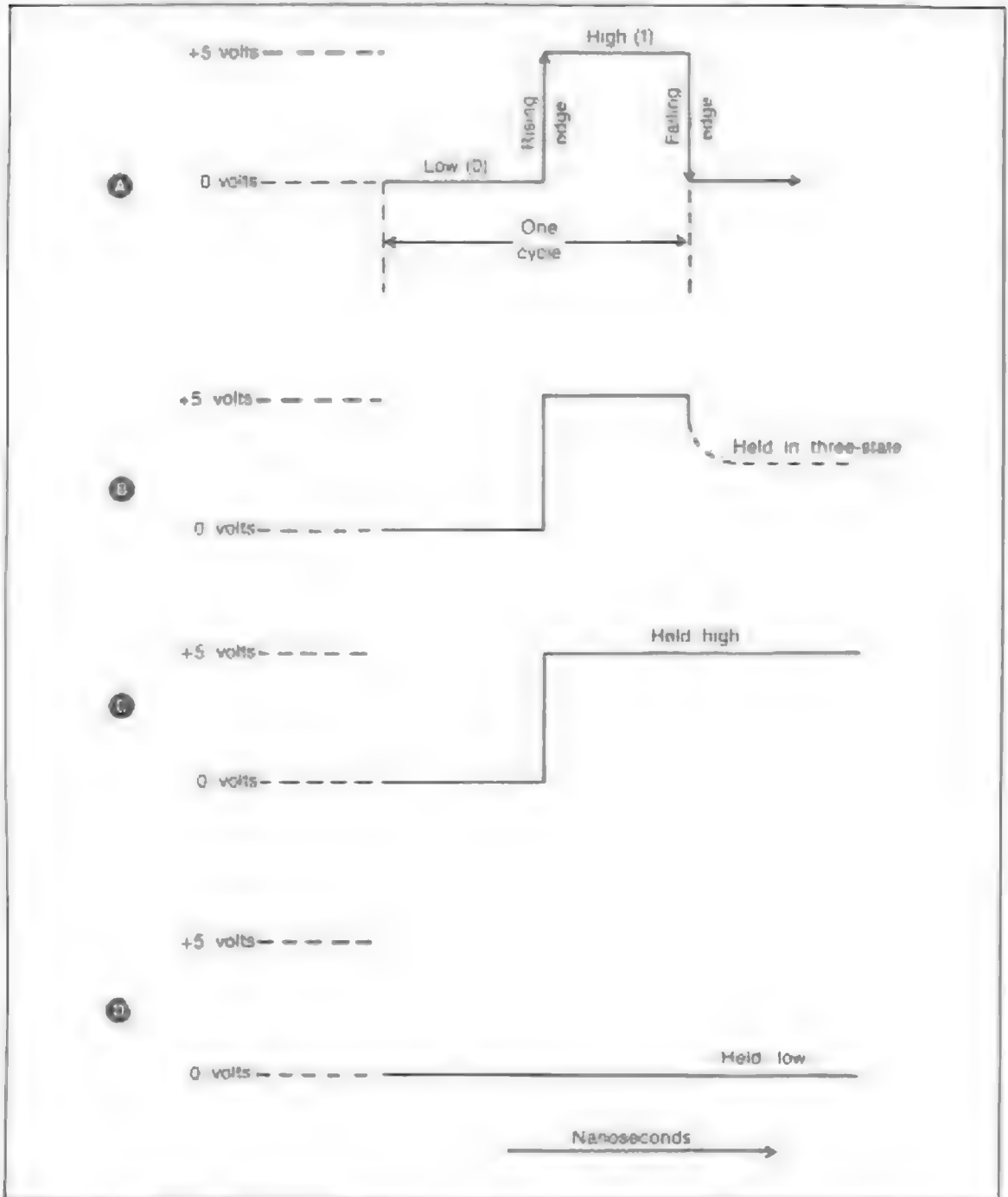
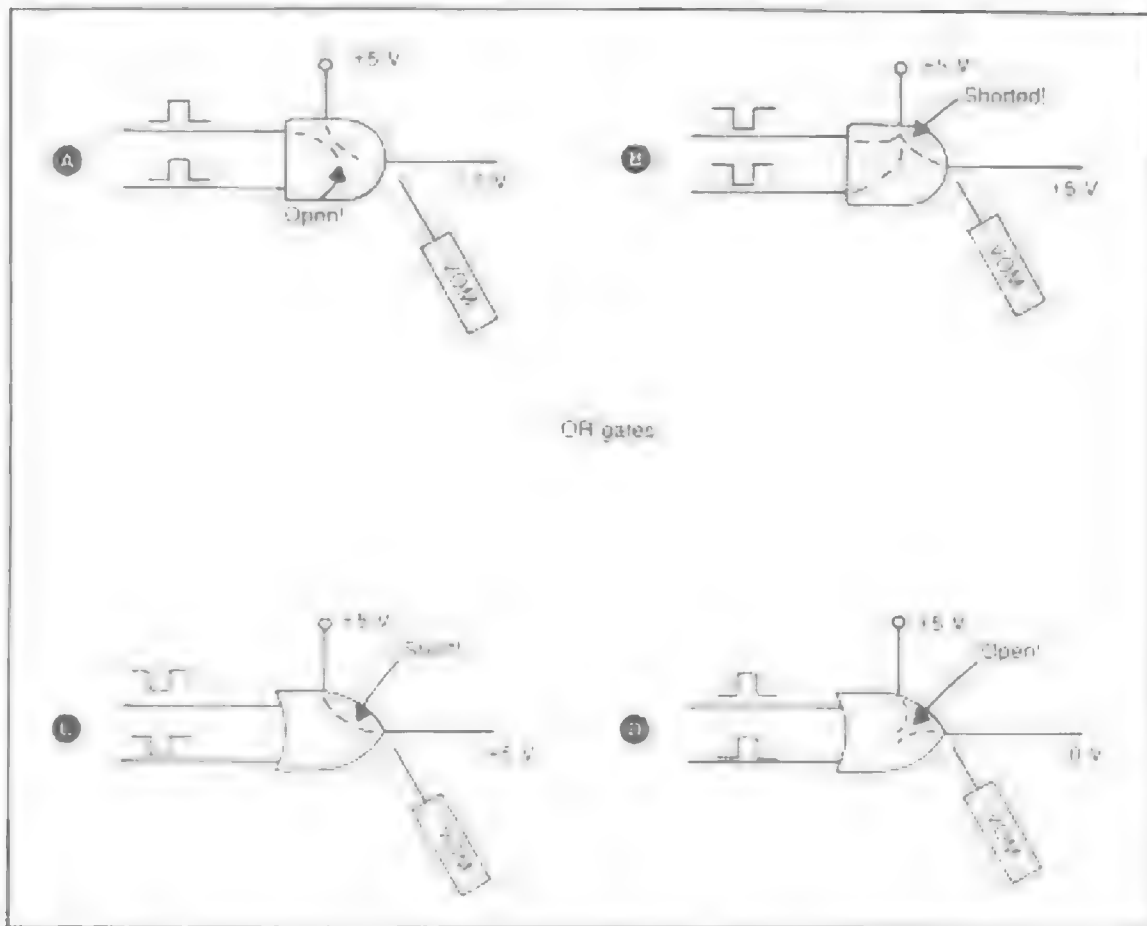


Fig. 10-3 The high and low states are graphed with voltage in the vertical plane and time in the horizontal. (A) changing state, (B) three-state, (C) high steady state (D) low steady state



OR gates

Fig. 10-4 If an AND gate should open, two highs could produce a low (A). If the same AND should short, two lows could produce output as a high (B). If an OR gate should short, two lows could produce output high (C). The same OR gate when open could output a low even though two highs are input (D).

way the gates work. On a normal schematic, since the internal wiring is not accessible, the entire YES gate is drawn as a triangle. The input lead is on a flat side and the output is at the pointed end opposite to the input.

All gates have a truth table. Truth tables got their name from the engineers that were using true and false to describe the high and the low states. If they had been using high and low, the tables might have been called the high table. Technicians think of high (H) and low (L) when they see a truth table. The truth table might contain any of the logical state descriptions shown in Fig. 10-7.

The table is a handy test medium. The columns are labeled Input and Output. If the input is an H in a YES truth table, the output is an H. When the input is an L, the output is an L. Therefore, if you are testing a normal YES gate with the vom or logic probe, your test reading on the input should also be found on the output. The truth table is actually a voltage-state test table.

The three-state control opens or closes the YES gate. The three-state controls are used extensively in the Commodore 128. They are used to control YES gate buffers and other chip circuits. Figure 10-8 reviews the three logic states, high, low, and no

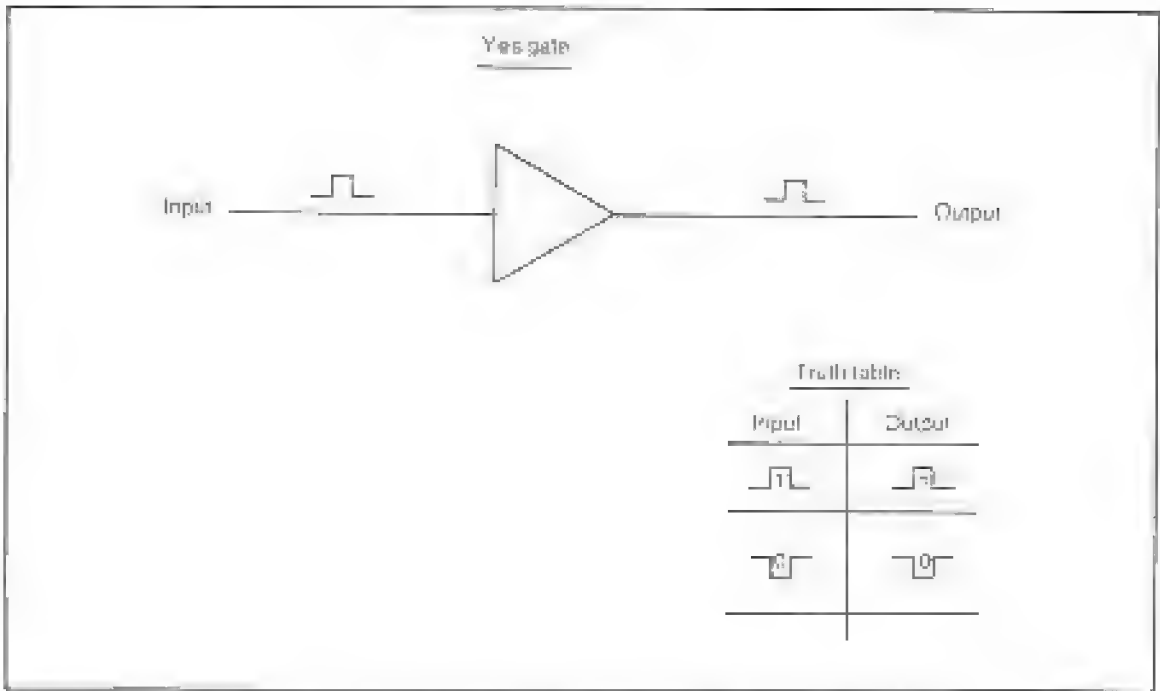


Fig. 10-5 The YES gate is so named because the output maintains the same state as the input.

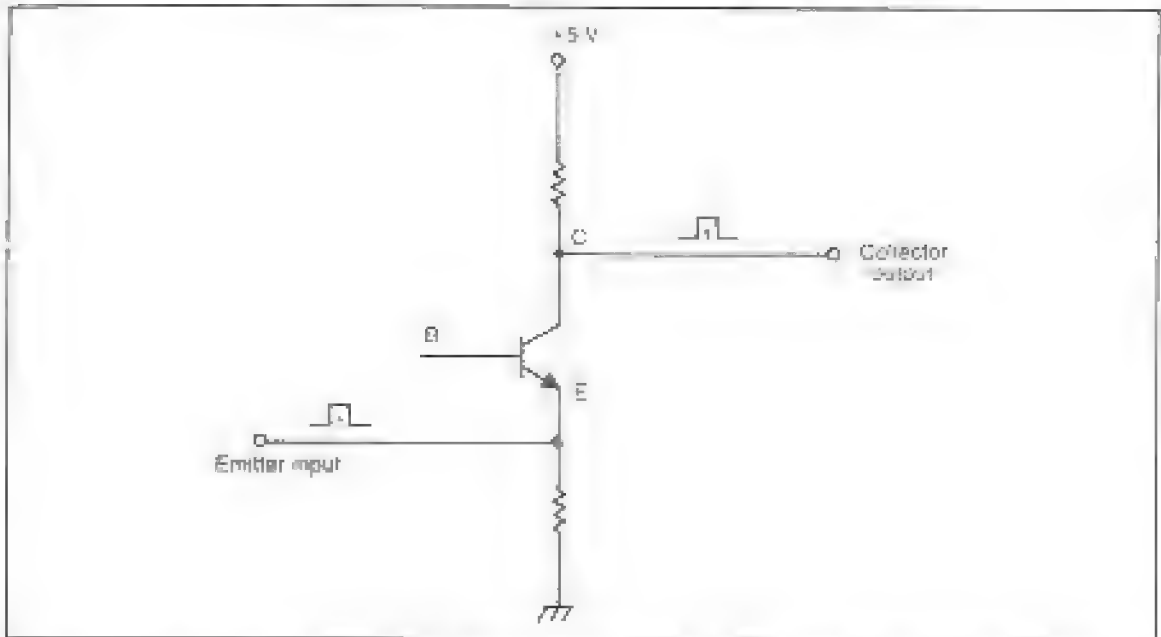


Fig. 10-6 An npn transistor with an emitter input and a collector output could be a YES gate. Whatever state is input will appear in the output.



Logical 1	Logical 0
True	False
High	Low
H	L
+5 V	0V
Set	Clear
Set	Reset
Yes	No
	

Fig. 10-7. Here are nine variations on the descriptions of the two logic states. The logic states, regardless of the name, are still only voltages.

state. The state of no state is vital to the operation of the computer.

For example, the address lines connect to every location on the memory map. The address that leaves the Program Counter and arrives on the 16 bus lines must open up only one address. If more than one of the addresses are read or written to, the computing will crash.

To avoid this, all the locations in the computer can be equipped with their own YES gate that also has a three-state control. While the computer is not addressing a location, all the three-state controls are off and the memory map cannot be accessed. As soon as an address goes out over A15-A0 the one location that is addressed has the three-state control in its YES gate go on. That way only the one location is contacted and all the rest of them stay in an inaccessible third state. There are a lot of three-state devices in the C128. The 8502, the

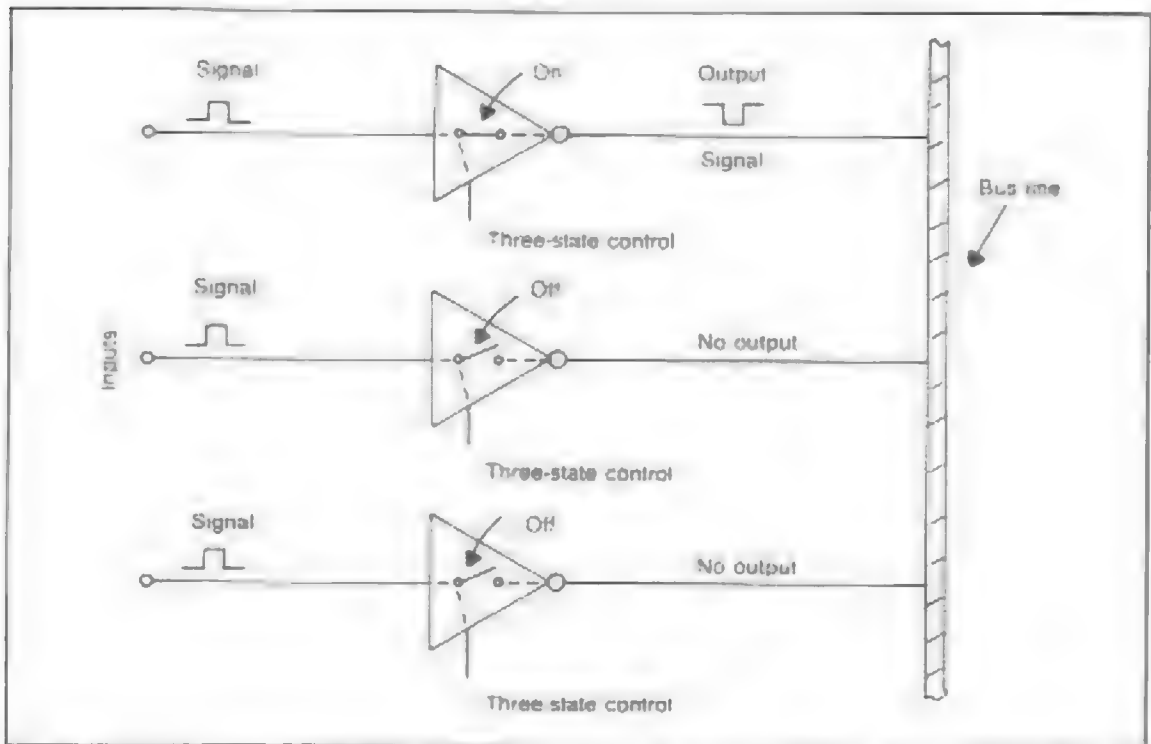


Fig. 10-8. The third logical state, when the chip is purposely turned off, is called three state. It is vital to keep all circuits off except the one in use during an operation so they do not interfere with each other.

RAM, the ROM, the gates in the address and data bus lines, and others have three-state capabilities.

NOT Gate

The NOT gate in Fig. 10-9 looks a lot like the YES gate on a schematic drawing. It has the same triangle shape and the same input and output leads. The only difference is a tiny circle between the pointed end and the output lead. This is called the *NOT circle*. The NOT circle is what makes the device a NOT gate. Whatever state enters the gate, the opposite state leaves the gate. If a high is input, a low is output. Should a low go into the gate, the low is changed into a high and the high comes out. The truth table for the NOT gate is shown in the illustration.

Figure 10-10 shows the bipolar circuit in a NOT gate. It is something like the YES gate. An npn transistor could act out the part of the gate. The VCC is still +5 volts. The obvious difference between the two circuits is in the input. The NOT gate input is connected to the base of the npn. The YES gate had the input tied to the emitter. There was no rever-

sal of logic with the emitter input. When the logic state enters the base, the state is reversed. The chip is also called an inverter.

The NOT gate or NOT circle is one of the most used abilities in a computer. The inversion of the logical state is the opposite or the complement. NOT gates always output the complement of the input. During testing, the fact that the output logic level is always the opposite of its input provides a quick check to test an inverter.

The 7405N chip in the C128 has six inverters. They are all used in output lines that need the state of the line complemented. This chip was discussed in Chapter 8. There are many other NOT gates and circles in a lot of the other chips in the 128. They are vital to the operation of the computer.

The NOT function is found in all sorts of places in the Commodore 128. As you peruse the schematic during troubleshooting, there are many terminals described with a straight line drawn across the top of the name. This overscore means NOT. This is the NOT line. It designates an inverted quantity. The asterisk I've been using is a substitute for the line.

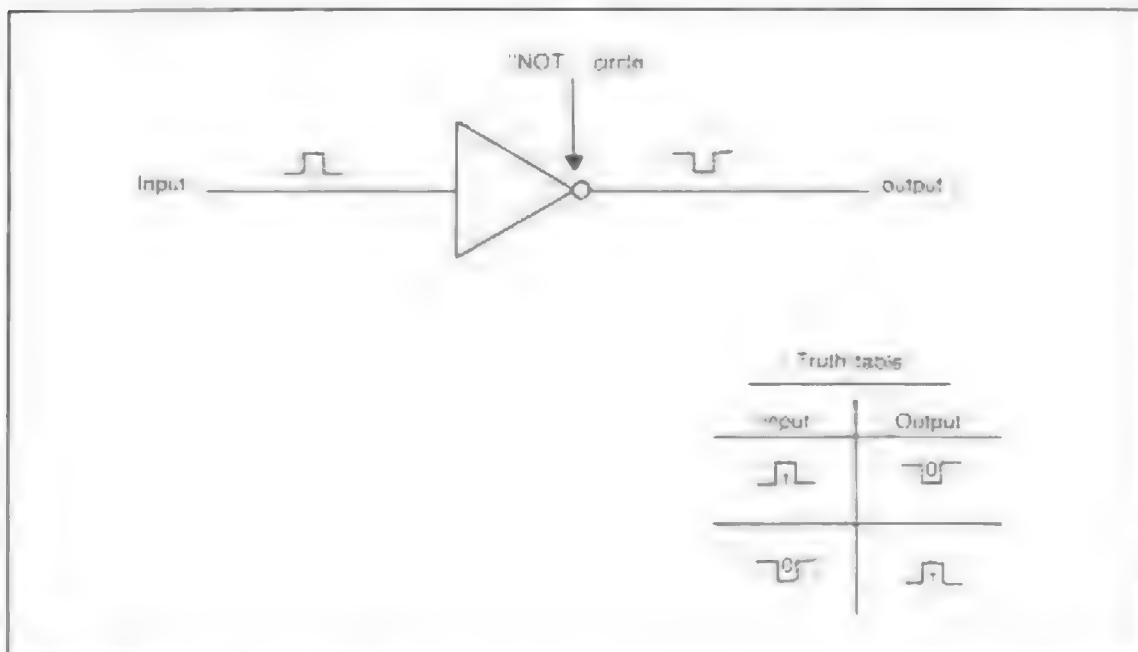


Fig. 10-9 The schematic symbol of the NOT gate looks like the YES gate except for the little NOT circle on the output point

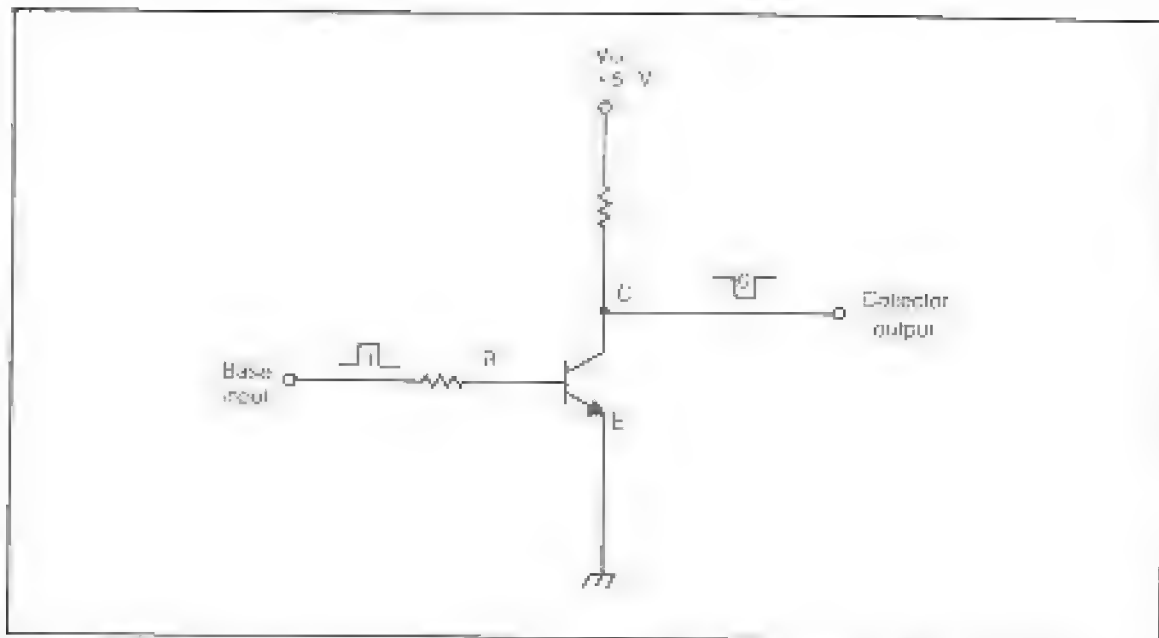


Fig. 10-10. The NOT gate can be constructed with the same npn transistor that the YES gate was made with. The input enters the base instead of the emitter and reverses the logic state at the output.

I use the asterisk because it is simpler to use in writing. When you see the asterisk instead of the line, it means the same thing as the overscore. It's a practical measure with no other meaning. The asterisk is also recognized as a NOT.

When a NOT sign is used on a terminal, it usually signals the type of logic state that will enable the terminal. For instance, if you see a terminal called *RESET, it means the pin is usually held high while the reset function is not being used. If and when the time arrives for the reset to be enabled, the terminal is injected with a low, and the reset goes into its routine.

On the other hand, if the terminal is called RESET without a line or asterisk, the opposite effects can be expected. The terminal will be held low when inactive. To enable the reset function a high is sent to the pin. The high turns on the reset routine.

When you are reading a schematic and you see a terminal name to describe its function, try to form the name in your mind. For instance, the read/write line in the C128 is described as R/*W. This would

be pronounced as "Read-NOT-Write." Actually the read/write line is usually held high in a read position. To produce a write the line must be forced low. Another example is the *IRQ line. It is an interrupt request. It is held high till the interrupt is required. Then the line is made to go low. A third example is RDY, a ready line. It is held low when it is on standby. To activate the RDY, the line is put into a high state. This form of thinking as you read the schematic should become second nature as you check out the test points on all the chips and connections.

AND Gate

The AND schematic symbol is shown in Fig. 10-11. It looks like a short fat bullet lying on its side. In contrast to the total of two leads on the YES and NOT gates, the AND gates in the 74LS08 chip of the C128 have three leads. It has two inputs and one output. Actually, an AND gate can have three or more inputs, but it will only have one output.

When the AND gate is also equipped with a NOT circle between the blunt rounded end and the

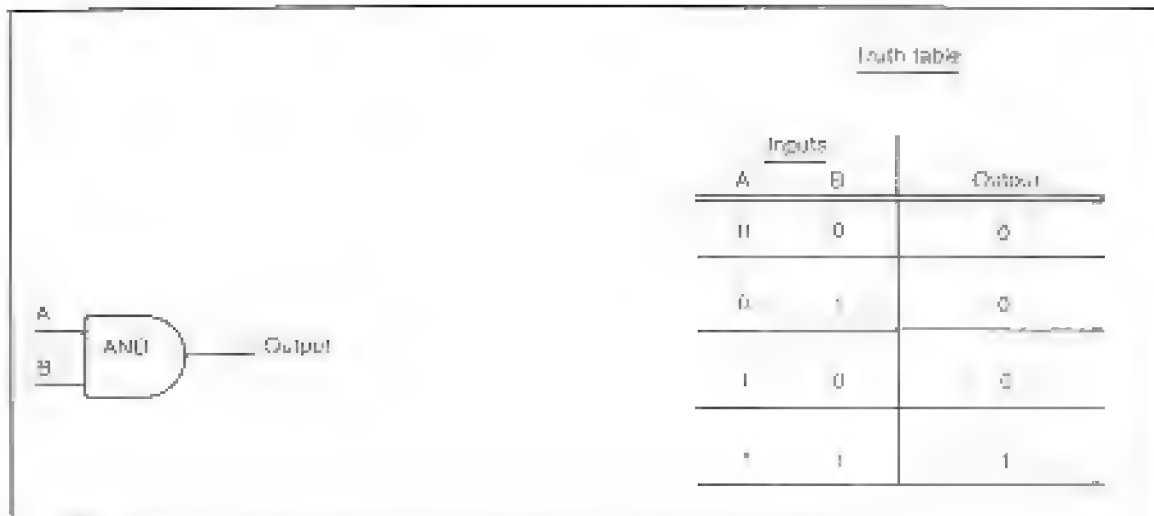


Fig. 10-11. The AND gate's output is low on all occasions except if both inputs are high

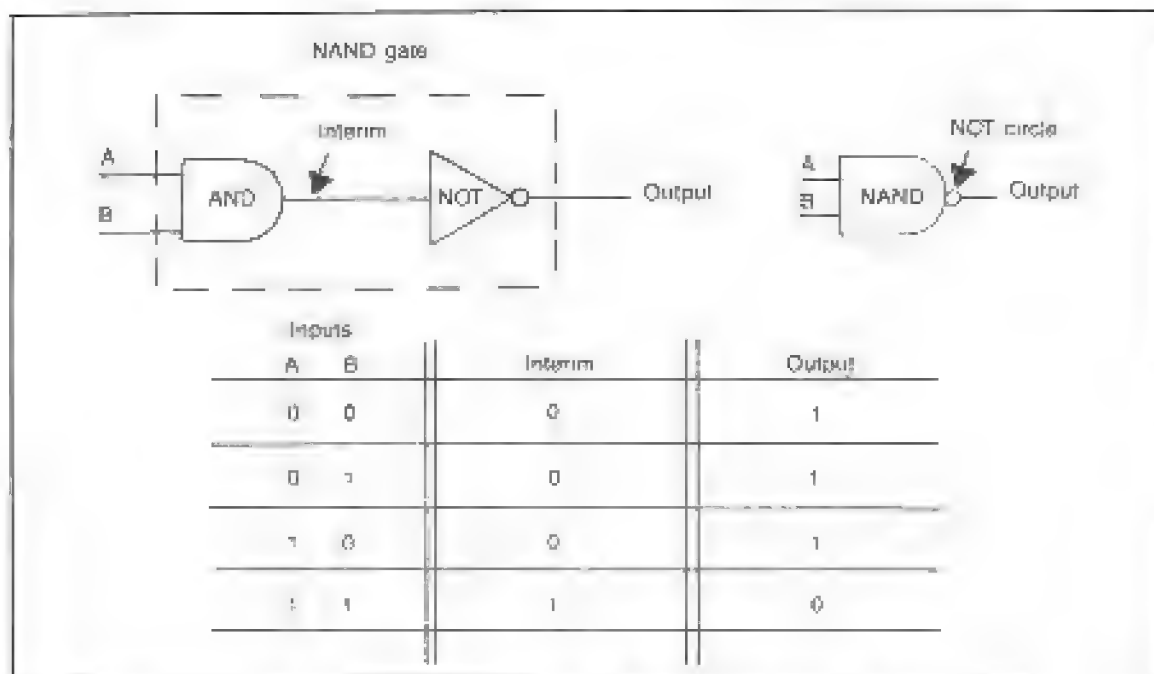


Fig. 10-12. The NAND gate is really a NOT AND gate. The interim output is that of an AND gate

output lead, as shown in Fig. 10-12, it becomes a NOT AND gate. You probably have seen these gates. The NOT AND is shortened to NAND. The NAND gate outputs are the complements of the

AND gate. There is an AND gate chip in the C128, the 74LS08. This chip was discussed in Chapter 8.

Inside the AND gate, as well as the other gates, there are a lot of microscopic components. There

could be all sorts of transistors, resistors, diodes, and so on. In fact, any gate can be formed by configuring a lot of other type gates on a chip and wiring them to tailor a particular gate. The little fat bullet on its side could be designating dozens of separate components.

From the servicing point of view, all you need to do is think of the gate in the simplest terms so that the testing and repair proceeds as rapidly as possible. That way you can trace a signal and pinpoint the source of a trouble. Except for your natural curiosity, it really does not matter which minute section of a minuscule internal transistor has given up the ghost. The important thing is to realize the total gate is dead and needs replacement. You can test the inputs and, from your understanding of the logic, be able to predict what the output should be. If the correct output state is present then the gate is okay. If the output logic is incorrect, then the gate becomes a suspect.

The classical description of an AND gate in electrical terms is a circuit consisting of an output load such as a light bulb with two switches in series. The

only way that the bulb in Fig. 10-13 will light is if both switches are closed. If one or both switches are open, the circuit is open and the lamp won't glow. There are four possible positions the two switches can assume. Assume that the energy is supplied by a 5 volt battery. The bulb has a high of +5 volts applied when both switches are closed and a low of 0 volts when a switch or both switches are open. We can call an open switch L and a closed switch H. The possible inputs are the following:

Inputs	Results
L-L	No light
L-H	No light
H-L	No light
H-H	Light

The same type of events take place when there are three AND inputs instead of two. The only difference is, three inputs create eight possible combinations that can be applied to the gate. Out of the eight inputs only one input group will light the bulb. That

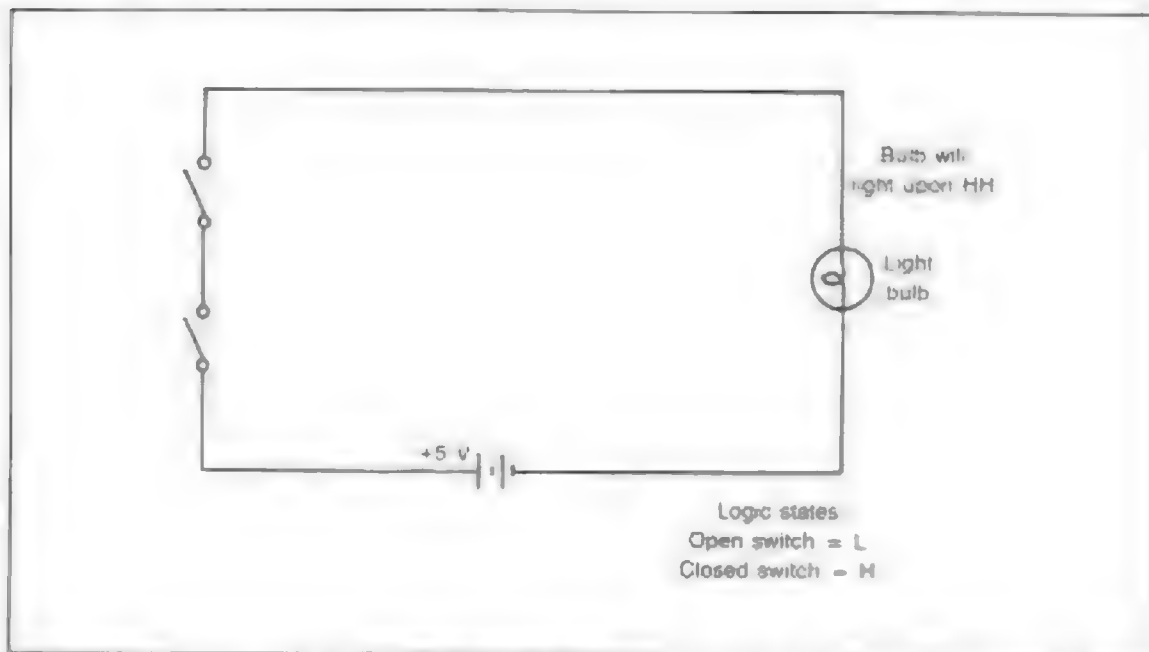


Fig. 10-13 The classical description of an AND circuit is made up of a battery, a light bulb, and two switches in series. An open switch is thought of as an L and a closed switch as an H. Only the application of HH will light the bulb.

is when all three switches are closed. The inputs and results look like the following:

Inputs	Results
L-L-L	No light
L-L-H	No light
L-H-L	No light
L-H-H	No light
H-L-L	No light
H-L-H	No light
H-H-L	No light
H-H-H	Light

When the AND gate has four inputs, there are 16 combinations. Five inputs makes 32 combinations of possible logic states, and so on. No matter how many inputs there are though, the only way the AND gate will output a high is if all inputs are Hs. When you are taking the state of the AND output, a high means all inputs are high. If one of the inputs are

low and the output is a high, that could be a symptom of failure. The gate could have shorted or opened in a way that is causing the unexpected high reading.

The actual AND circuit could be based around a pair of pnp transistors wired in parallel. Refer to Fig. 10-14. The emitters of both the pnp's connect to +5 volts through a resistor. The AND output is from the emitters. The two AND inputs are entering through the two bases.

When either or both inputs receive a low, one of the transistors or both will conduct and the output will be low. The only way a high can emerge from the output is if both inputs are high. When that happens neither pnp can conduct and the output emitters rise to the supply of +5 volts.

You could think of an AND gate as an electronic combination lock of sorts. A low is output while the lock is shut tight. The only way the lock will open is when the correct combination is input. The correct combination is the input of all Hs.

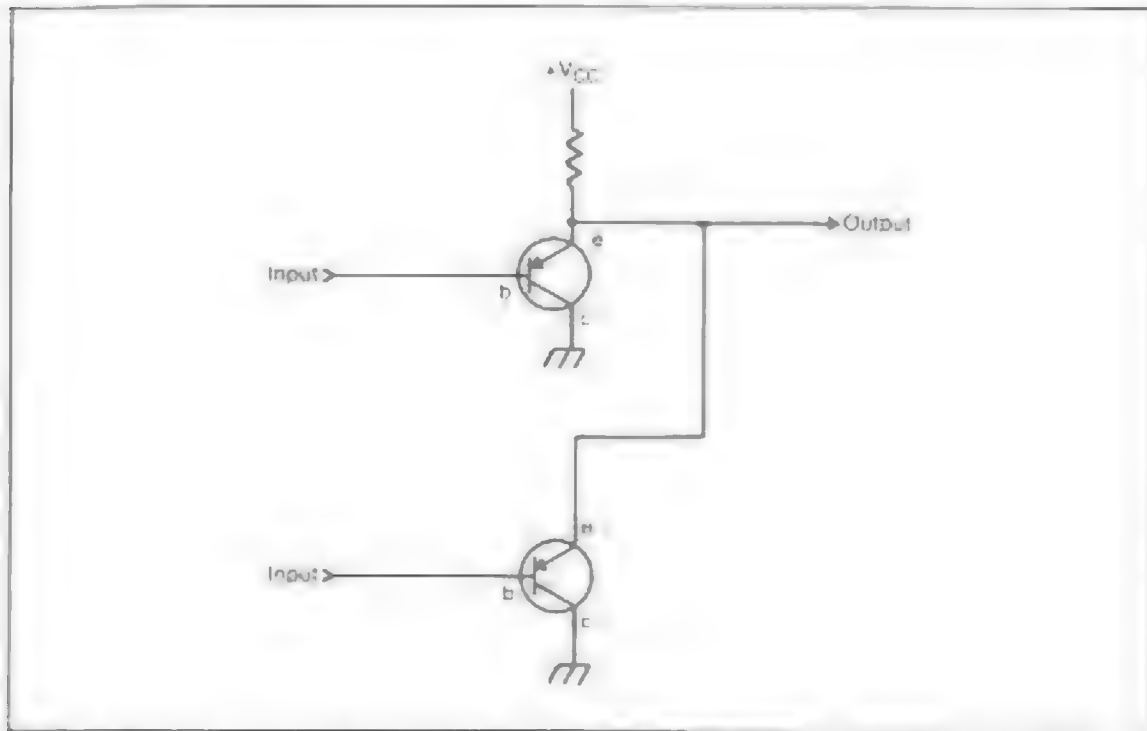


Fig. 10-14 Two pnp transistors wired in parallel with a base input and an emitter output can be an AND gate.

OR Gate

Figure 10-15 shows that the 74LS257 chips use OR gates as part of the internal wiring. The OR gate outputs are connected to pins but the inputs are attached to the outputs of the AND gates that they are doing the multiplexing with. The inputs from the address bus are the inputs to the AND gates. This chip was also discussed in Chapter 8.

The OR gate is drawn schematically like an artillery shell on its side, instead of a fat bullet like the AND gate. The classical electrical representation of the OR gate is also shown as switches that operate a light bulb. The wiring in Fig. 10-16 is different than the AND circuit. Whereas the AND circuit had switches in series, the OR circuit has the same

switches in parallel. The AND circuit also needed all the switches closed to light the bulb, but the OR circuit only needs a single switch closure to illuminate the bulb. The possible inputs are the following:

Inputs	Results
L-L	No light
L-H	Light
H-L	Light
H-H	Light

The OR gate also can have more than two inputs. There can be three, four, or more. Like the AND gate, two inputs produce four input possibili-

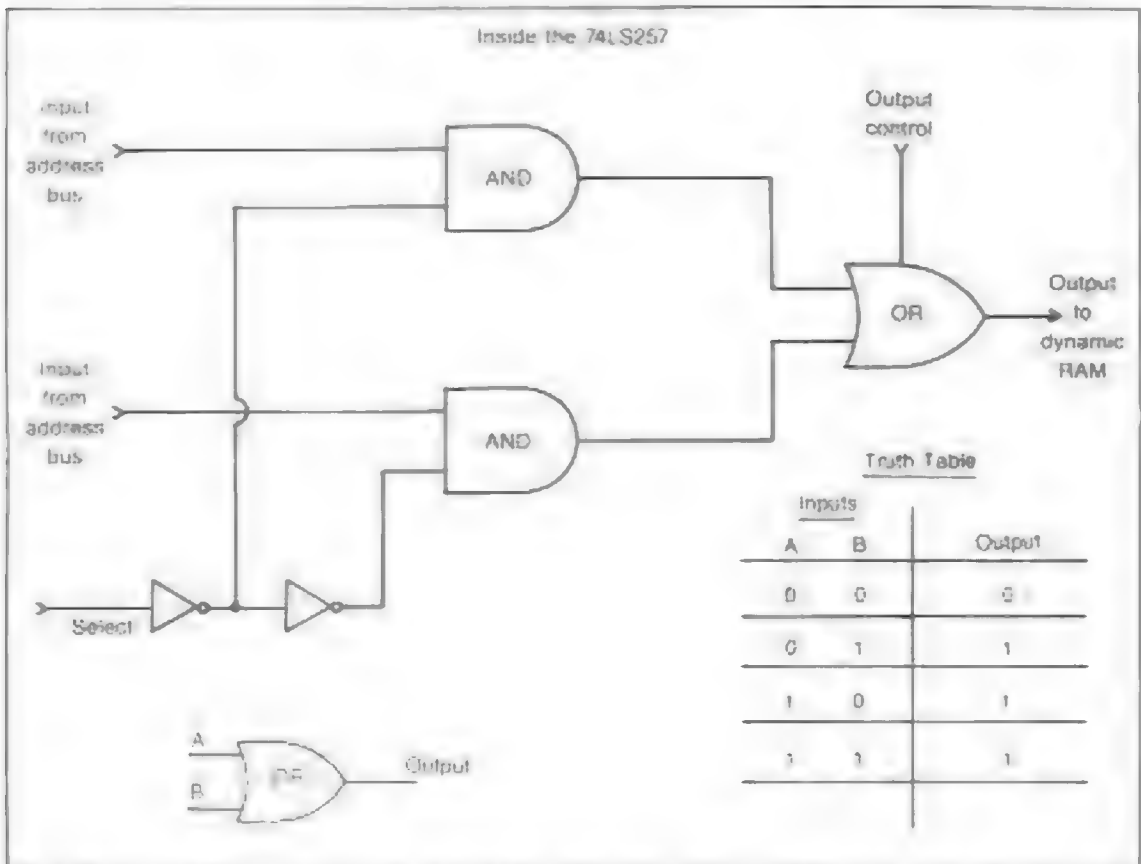


Fig. 10-15. The OR gate on one output pin in the 74LS257 receives signal from two AND gates. Each AND gate is operated by one address bit and a select signal. The OR gate symbol looks like an artillery shell lying on its side. The OR gate's output is high unless both inputs are low.

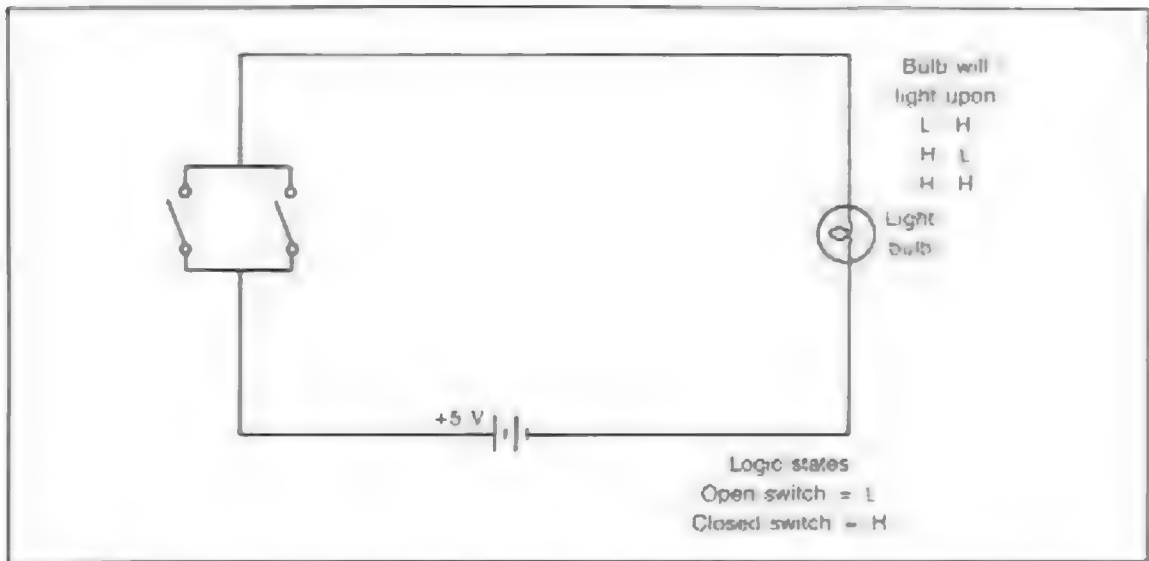


Fig. 10-16. The OR gate can be thought of as two switches in parallel. The bulb will light if either one or both of the switches are closed.

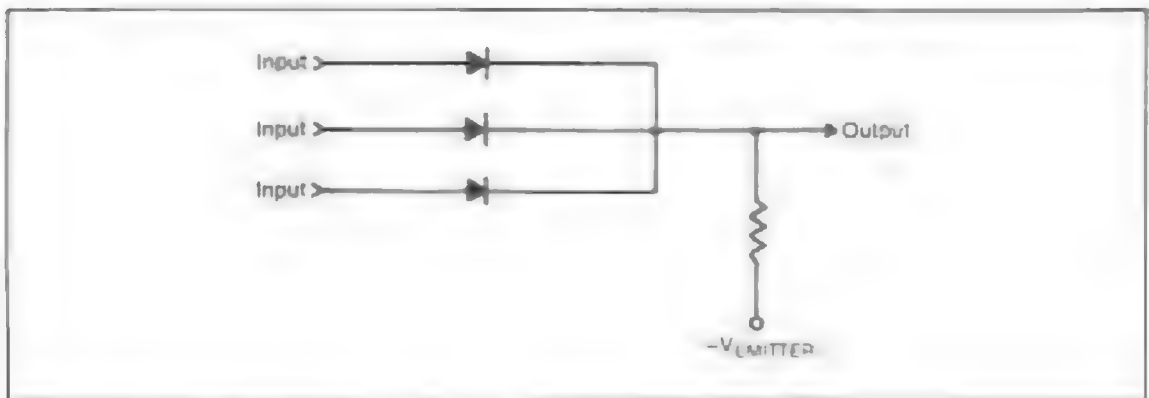


Fig. 10-17. The OR gate can be built with diode inputs in parallel.

ties, three inputs can have eight combinations of Hs and Ls, and four inputs make 16 possible ways that the inputs can be arranged. However, no matter how many inputs, the OR gate will output a low only when all inputs are low. Otherwise the OR gate will output a high. If just one input out of many is high and all the rest are low, the OR gate will put out a high.

An OR gate can be built with a number of diodes and a resistor as shown in Fig. 10-17. If you want a two input gate, two diodes are used. For a

three input gate, three diodes are needed. One diode is needed for each input.

The diodes are wired in parallel. The anodes of the diodes are the input pins and the cathodes are tied together. A resistor from -5 volts is connected to the cathode junction. The junction is also the OR output pin.

While the inputs are all near 0 volts, the diodes do not conduct and the output is held at -5 volts which is a low. If a high should appear at one of the

anode inputs, that diode conducts. This makes the cathode junction rise up 5 volts to a relative high. A high on any of the anodes will also produce a high at the output. This is logical OR activity.

Exclusive-OR Gate

The C128 does not use any exclusive-OR abbreviated to XOR, chips. The XOR function though is very important to the machine. One of the instructions that the 8502 will respond to is EOR, which is also a way to describe the action. EOR is the acronym for exclusive-OR. When you order the 8502 to EOR, it will exclusive-OR all the bits in the accumulator register with any memory location you provide. There will be more about this in the next chapter on registers. Meanwhile, we will review the XOR gate so you will understand the EOR when you get to it.

The XOR gate symbol in Fig. 10-18 looks almost exactly like the OR symbol. The only difference is a space between the input leads and the bottom of the body of the symbol. A closer look reveals that at the input end, the leads are attached to a curve and there is a space between the curve and the rest of the symbol. If you think of the symbol as an artillery shell, then you can think of the input leads as the ramrod used to push the shell into

the big gun. The output lead is on the pointed end of the shell appearing symbol.

Let's compare the OR and XOR inputs and outputs to see the difference the exclusiveness makes. If you are signal tracing a two-input OR or an XOR gate there are four input combinations possible for both gates. They are the same inputs whether the gate is an OR or XOR. The difference is in the outputs.

On the OR output, the state will always be a high except if both inputs are lows. If you logically think about these results, they do not really follow the definition of OR. The OR definition is, if one 'OR' the other input is high, then the output is high. This is true for L-H and H-L. For L-L, neither input is high so the definition indicates a low, which will be present as long as the gate is okay. For the last possibility though, both outputs are high, H-H. Yet the output is high.

The XOR gate is a variation of the OR gate that does follow the logic. When the four input possibilities are considered, they are identical to the OR gate except for the last possibility, H-H. When H-H are the inputs of the XOR gate, the output is a low. The XOR gate only outputs a high when one or the other of the inputs are high. If both inputs are the same, both low, L-L, or both high, H-H, the output is a low.

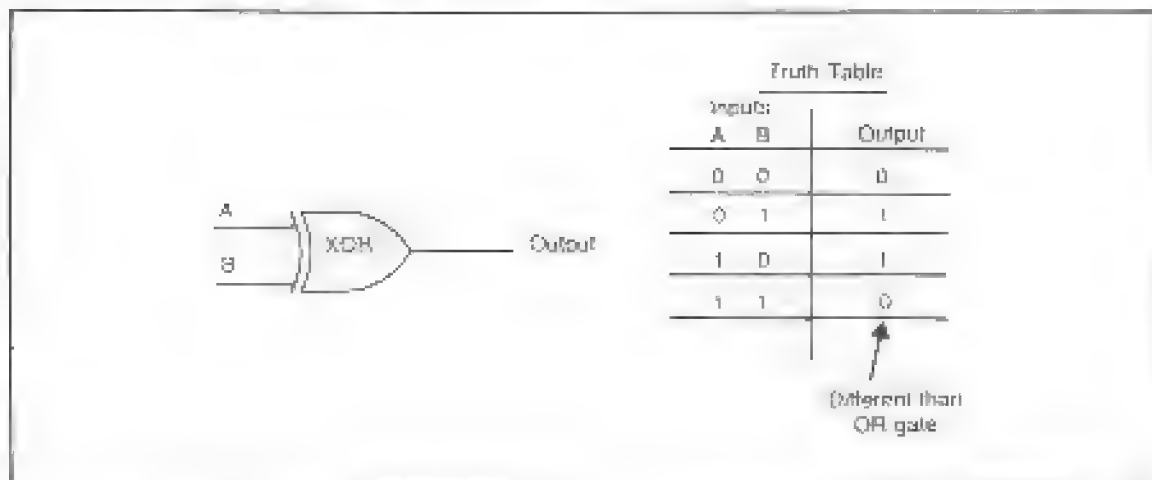


Fig. 10-18. The XOR gate is similar to the OR gate, but when two highs are applied to the inputs, the XOR outputs a low.

NOR and XNOR Gates

You will not have to contend with NOR, XNOR, and other gate variations. The C128 deals mostly with AND, OR, NOT, and some XOR logic abilities. If you learn these four functions and understand what the outputs of the gates do with various inputs, troubleshooting the C128 will become appreciably easier. I include a brief description of these other gates simply for completeness.

NOR is NOT OR and XNOR is exclusive NOT OR. The NOT simply changes the outputs to a complement. The output of the NOR gate is the complement of the OR output. The same thing goes for the XNOR. Figure 10-19 shows the complementary relationships. Compare these with Fig. 10-15 and Fig. 10-18.

GATE TESTING TECHNIQUES

When a gate is to be tested the best equipment is the vom and the logic probe. These instruments however, can only give you the surface indications. As we have seen, the chips are circuits within circuits within circuits. Deep down at inaccessible levels are the bipolar transistors, the FETs and other components that actually do the computing.

They are forever sealed at manufacturing in their microscopic space. You can think about them, but under normal servicing circumstances you'll not encounter them directly.

A gate is at the next level of size and they do have test nodes that you can measure and obtain readings. The gate circuit level is the level where most of your voltage and logic state testing will take place. There are inputs, outputs, VCC (collector) and VDD (drain) pins readily available. The more you know about the way the Hs and Ls progress through the gate level circuitry, the quicker you will spot incorrect logical states and pinpoint troubles.

The top level of circuitry is the chip itself. When the chip is considered as a single component and is replaced as a test, you are in the overview mode. This is the first approach to a repair and is the gist of the chapters up to this one. More than 50% of actual troubleshooting that takes place in the field is in this overview mode. As a matter of fact, manufacturers have taken this mode of servicing a couple of steps further. They often will replace the entire board or even the entire computer to get you back in operation. The troubled computer can then be placed back on the factory production line and be recycled back to new.

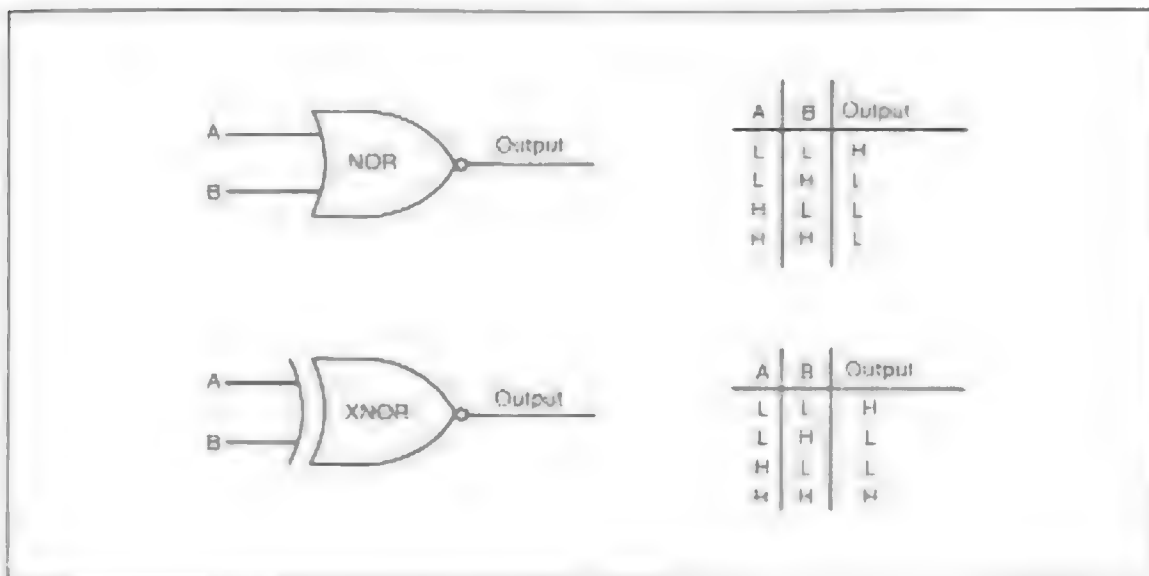


Fig. 10-19 The NOR and XNOR symbols and truth tables

Independent servicers, without the benefit of a large factory, follow the same procedure they have been using for years on TV repairs. Computers are easily carried. The home computer or small business computer owner disconnects a broken unit and carries it into a repair shop. The servicer approaches the repair by changing the socketed chips that are indicated by the symptoms of trouble. Once the socketed chips are deemed okay, then the servicer

will pull out service notes on the machine and attack test nodes with the vom or logic probe. He looks for incorrect voltages and wrong or missing logic states: missing or wrong supply voltages, highs where there should be lows, lows where highs should be present, three-stating in the wrong places, and so on. As you can see, knowing the way gates process highs and lows is a must.

11. Servicing Digital Registers

By this time, you should know that, besides the logic gates, digital electronics requires another entire category of TTL and MOS devices. They are called registers. Registers are found throughout the digital circuits. There are many in all the main LSI chips. There are thousands upon thousands of them in RAM and ROM. Every resident of the memory map is a register of one sort or another. A number of the rest of the smaller chips are forms of registers.

The important difference between a gate and a register is illustrated in Fig. 11-1. A register can store a high or a low while a gate cannot. As soon as a logic state arrives at a gate, if it is not three-stating, the high or low gets a quick passage through to the output. It can't linger in the internal circuit. It is forced on through. The only time required is the propagation delay of the electron movement, which is typically about 15 nanoseconds. Besides this

15 billionths of a second pause, the state passes through the gate.

A register on the other hand has bit holders. It is a place of storage. Once a high or low enters a register bit holder it can stay as long as you want it to, providing the electricity stays on.

A register is composed of one or more single bit containers. In the 4164 dynamic RAM chips, there are 65,536 single bit registers on each chip organized as $65,536 \times 1$. The 2016 Color RAM chip is organized as 2048×8 . This chip has 2048 8-bit registers in its memory matrix. The 7415373 D Latch connected to the Color RAM has one 8-bit register. The Program Counter in the 8502 is a 16-bit register. All the various sized registers do the same type of job. They are all able to store bits according to their organization.

Each form of register except the dynamic RAM uses flip-flop circuits as bit holders. The dynamic

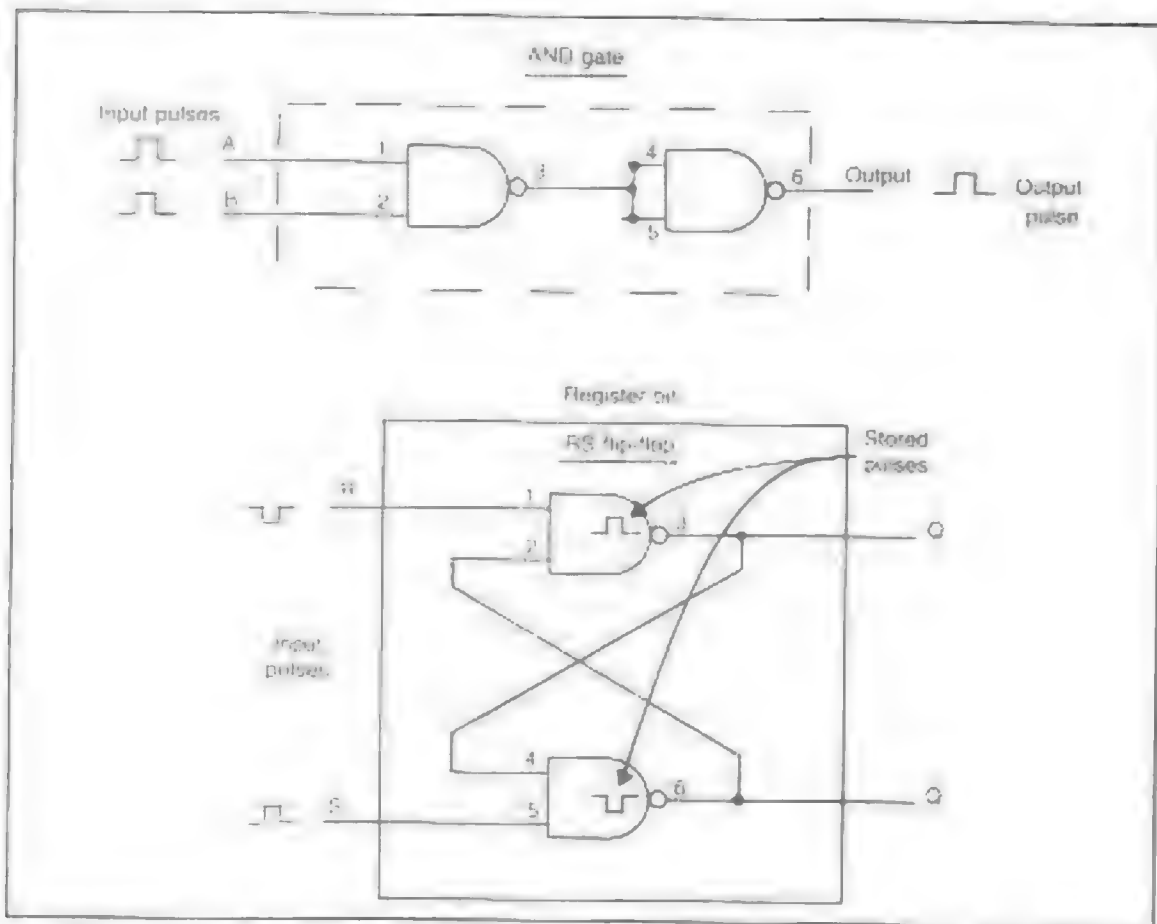


Fig. 11-1 The main difference between a gate and a register is that a register can store a logic state and a gate cannot. In the top circuit, two NAND gates are wired together to form an AND gate. Inputs are quickly ushered from the inputs to the output. The bottom circuit shows the same two AND gates wired as a register bit. Inputs can be stored in the internal circuitry.

RAM uses the voltage that is contained in a capacitor charge. Chapter 8 contains additional information on each of the following chips.

FLIP-FLOPS

The flip-flop circuit is as old as electronics. It is a means of storing a charge. In the digital circuit, the presence of a charge is a high, and the lack of a charge is a low. With a flip-flop, you can not only store the charge, but you can manipulate it. You can change a high to a low or a low to a high. This is

a very valuable ability. When you couple the storage and the state changing with the basic gate logic of NOT, AND, OR, and XOR, you are computing.

A flip-flop can be built by crisscrossing two triode vacuum tubes, or two npn's, or two pnp's, or two FETs, or even two gates. They are all able to characterize a flip-flop. Let's examine the step-by-step flip-flop action of the circuit in Fig. 11-2.

The circuit shows the transistors with inputs into their bases. Since the inputs are into the bases, the individual pnp's are wired as inverters. The

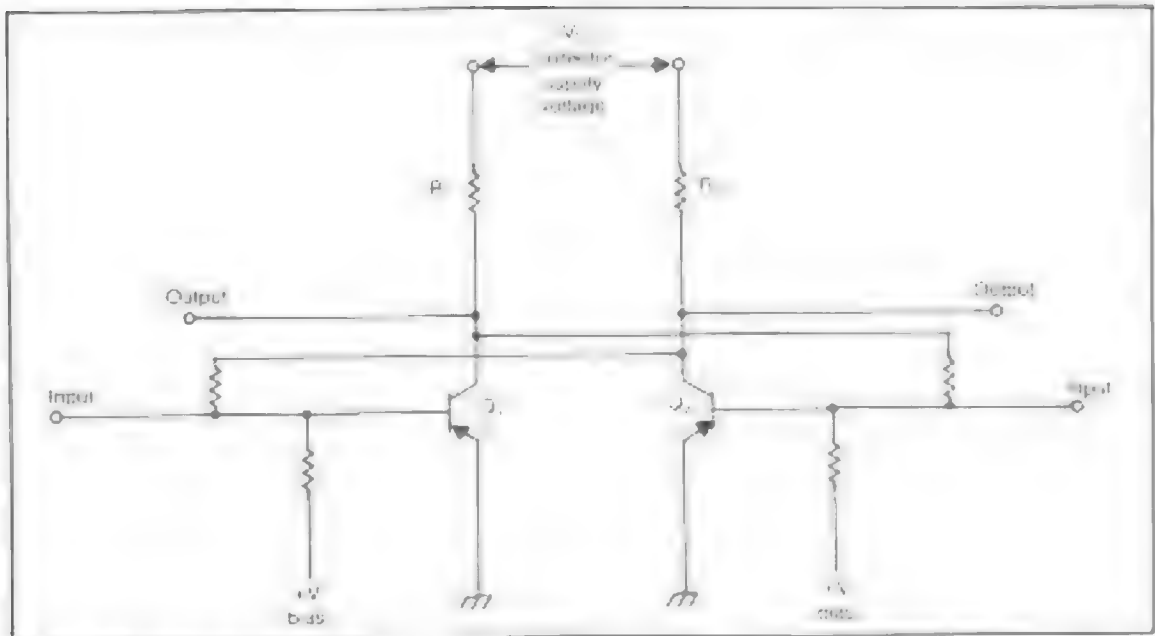


Fig. 11-2 The basic flip-flop can be built around two pnp transistors.

bases are then cross-coupled into the other pnp's collector. When you power the two pnp's, they both try to conduct. However, one is always slightly quicker than the other. There is feedback from the collectors to the other base. The quicker pnp will go into saturation. The slower one will thus be forced into cutoff. They will hold this state of conduction.

The one in saturation outputs a high as the pull up resistor has a large voltage drop across it. The one cutoff outputs a low since its resistor has no voltage drop and is at the $-V_{CC}$ supply voltage. The two transistors will maintain this state. The only way it can be forced to change is if a high pulse is applied to the base of the cutoff pnp or a low pulse is sent to the base of the saturating pnp. If either event takes place the two transistors will flip-flop their logic states. The pnp that was cutoff will then saturate. The pnp that was saturating will cutoff. The flip-flop will then hold that state till another pulse comes along to change it again.

While you can't get into a chip to test flip-flops, it is useful to understand the way it stores a voltage state. There are a number of tests you can make on flip-flop circuits to see if they are able to store

voltages. For instance, if you know the address of a suspect register or bit holder, you can use the BASIC PEEK function to see if it is actually holding the highs and lows that it is supposed to. The POKE command can be used to install test bits into the register and then PEEK can look to see if the test bits ever arrived. Refer to Fig. 11-3. There will be more about this in later chapters.

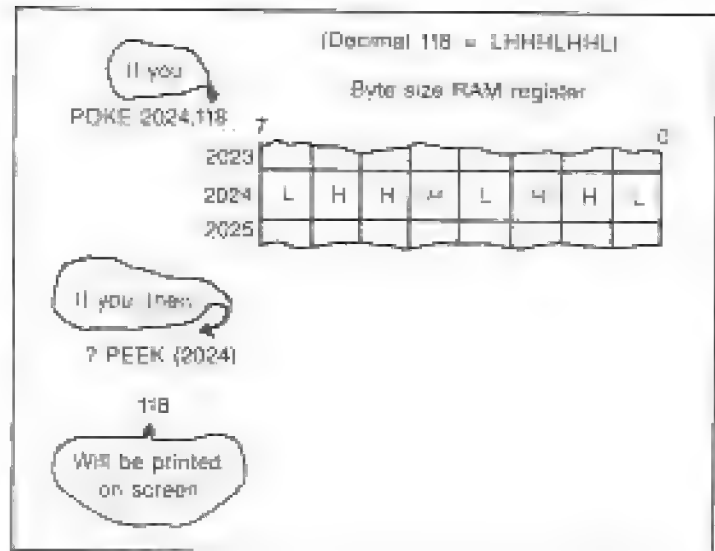
There are a number of chips in the C128 that are flip-flop types. The 74LS74 in the clock circuit contains two D flip-flops. The 74LS373 is called a D latch.

The 556 chip has two separate flip-flops inside. The flip-flops are completely independent of each other except for the fact that they share VCC and ground. The timing ability comes from some comparator circuits that are also built into the silicon. Let's examine the Commodore's chip flip-flops.

74LS74 D Flip-Flop

The 74LS74 chip contains two flip-flop circuits. The drawing of the pinout was shown in Fig. 8-14. There are two pins for VCC and ground and six other connections.

Fig. 11-3. Any register in the memory map and every bit in the register can be tested with POKE and PEEK tests.



The D FF has one Data input. In fact, that is why it is called a D. Other type FFs are called D Latches and RS. The D has one Data (D) input. The input is a single high or low at a time. When the logical level enters the D pin, it is stored in the flip-flop circuit.

When a low enters D, a low will appear at the Q output. Q is the important output. The other output, \bar{Q} , will become a high. This is not an important output. It can be useful sometimes, but the Q output is the one that usually passes the logic state.

The flip-flop, though, will not relinquish the state it is storing that easily. It must be triggered by a clock pulse that goes high. The FF will only give up its stored state when a high clock arrives. The 74LS74 requires a high clock pulse to pass the state from D to Q. Other FFs can use a low clock pulse for the data transfer.

The 74LS74 has two other input capabilities. They are Preset and Clear. They are used sometimes, but not all the time. When it is necessary to preset the Q output to a high, the input pin preset receives a low. When Q is required to be a low, then the low is inserted at the Clear pin.

74LS373 D Latch

The 74LS373 is a D latch. The D latch is a close relative of the D flip-flop. This latch has eight FFs

in it. Latches can come with four, six, or eight to a package. They are usually used in bus lines and different bus lines in different applications use different packages. (See Figs. 11-4 through 11-7.)

A latch is a parallel output device. With this eight pack, you can send eight bits over eight address lines and have them stored or latched till you want them to continue their journey. The latch is different than the D in that the D FF uses a clocked input to release the stored contents. A latch uses a latch enable pin. Pin 11 is the latch enable.

When data arrives at the eight FFs, it is stored. The latch will hold the data while the latch enable pin is low. As soon as a high arrives at the enable pin the data is freed and allowed to pass onto the Q output. In the C128, the data comes from two places. The data entering D5-D0 is an address from VIC. The data at D7 and D6 arrives from the multiplex chip. The output pins Q7-Q0 connect to SA7-SA0 of the address bus. The chip is helping out in the complex addressing chore. Both the MPU and the VIC, at different times must address RAM. The octal latch 74LS373 works with the multiplex chip and the VIC to supply the correct address bits at the right time.

Each of the eight flip-flops have a D input and a Q output. The D and Q numbers show which is which. For instance, D0 is the input to FF 0 and Q0

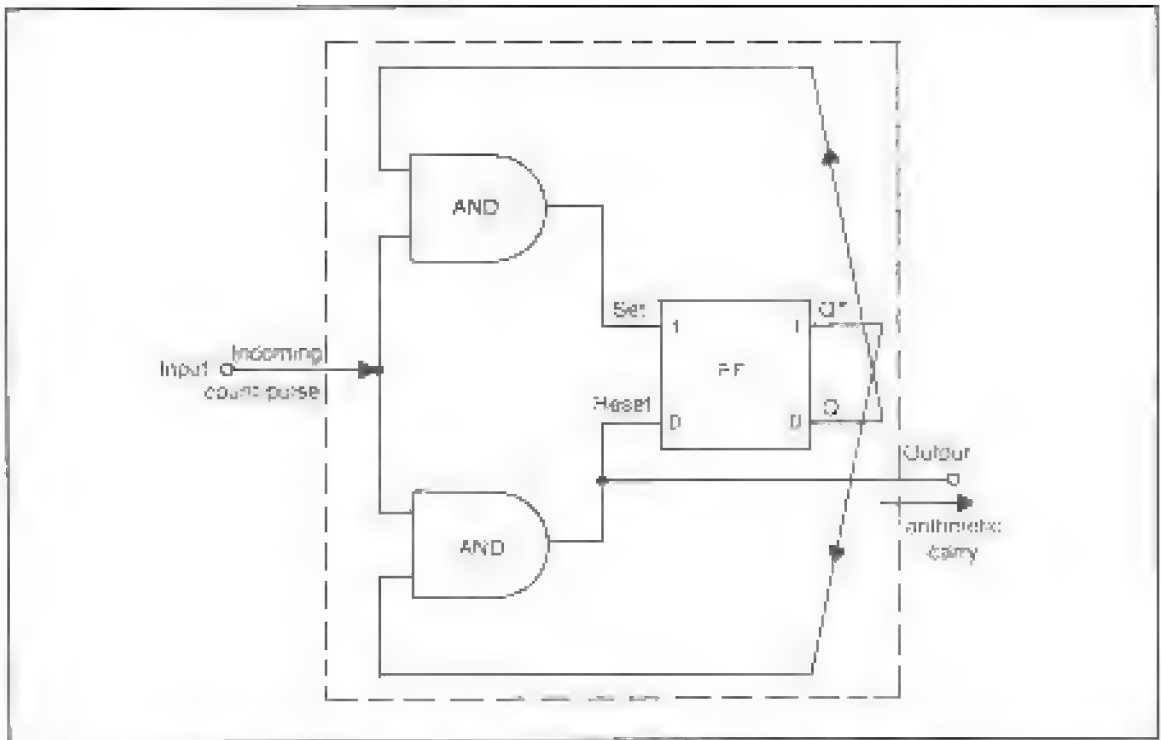


Fig. 11-4. The typical binary counter can count 0, 1. The output pulse can then be applied to the next stage as an arithmetic carry.

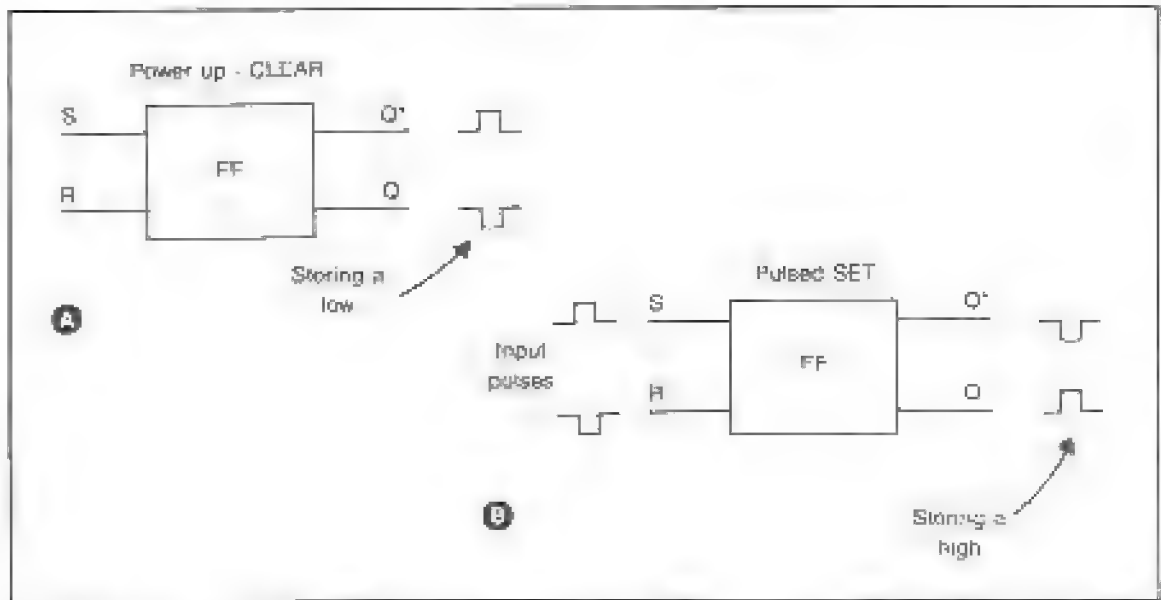


Fig. 11-5. In the RS Flip-Flop, the Q output exhibits the logic state. \bar{Q} always has the opposite state of Q.

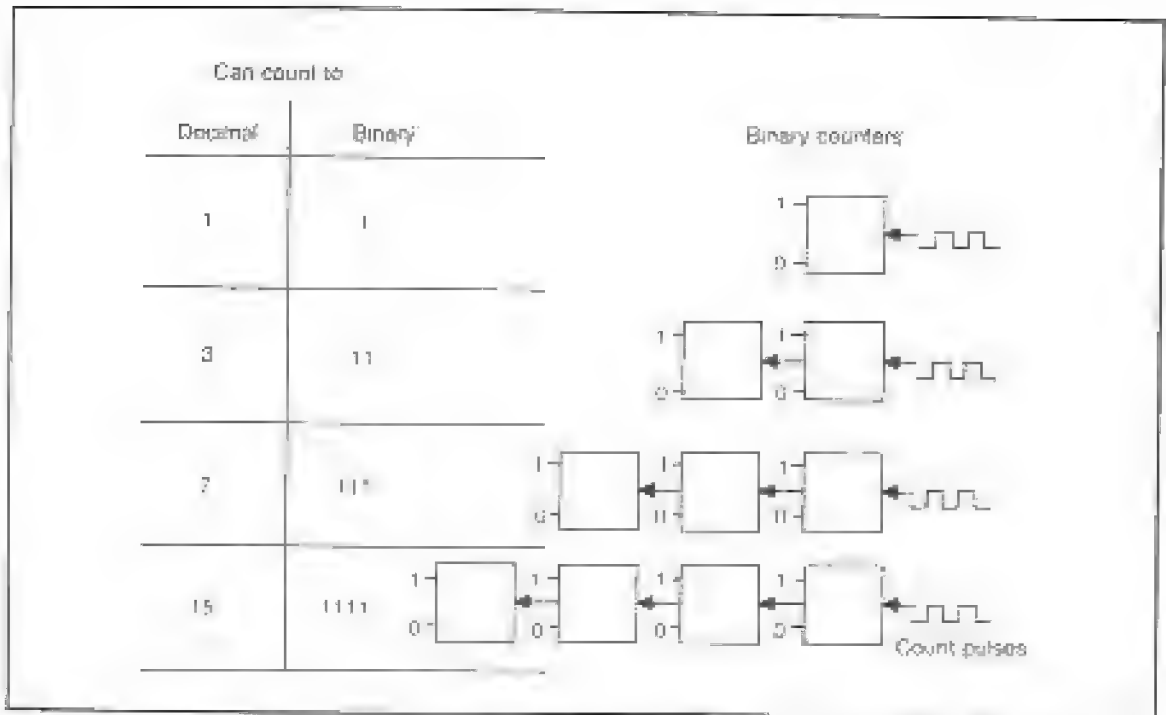


Fig. 11-6. Two counters can count 0, 1, 10, 11. Three counters can count 0, 1, 10, 11, 100, 101, 110, 111. Four counters can count 0, 1, 10, 11, 100, 101, 1010, 1011, 1100, 1101, 1110, 1111.

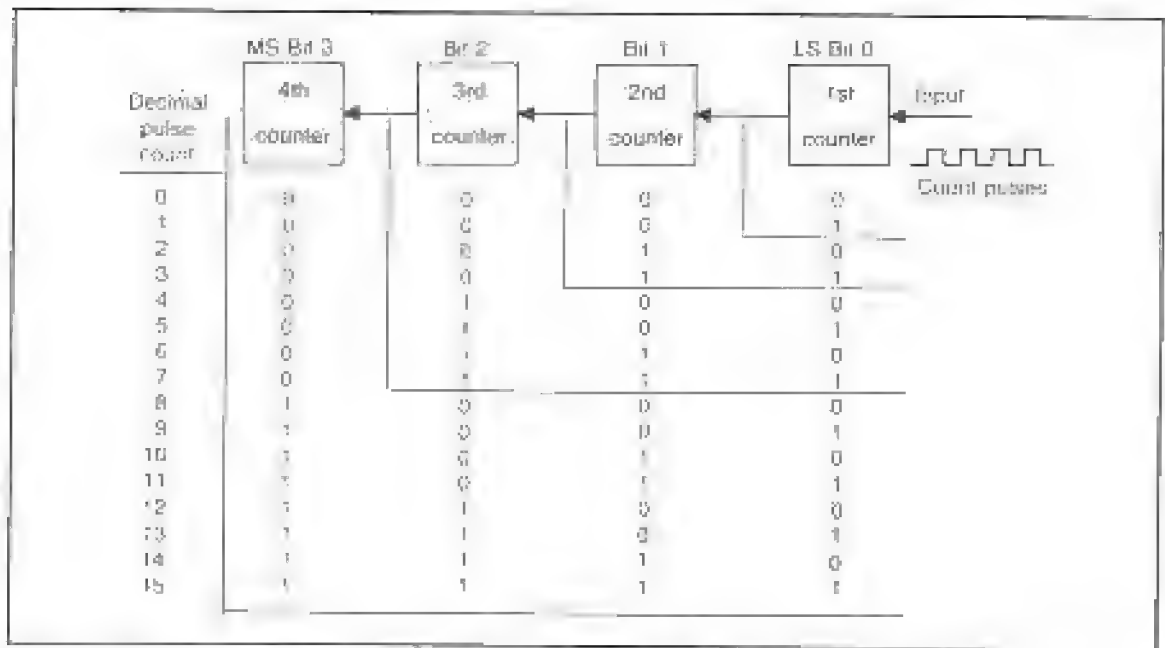


Fig. 11-7. Four binary digits are called a nybble. It takes the four bits to code one decimal digit.

is the output for FF 0. All of the flip-flops are tied together on one clock line and one three-state line. Pin 11 is the clock line input. It updates the latch with the strobe signal from the VIC (*RAS). It will open the latch and let the eight bits pass to the Q outputs. Almost at the same time, a new set of eight bits moves in and takes the place of the bits that just left.

Pin 1, *OE is the three-state control for the entire chip. The signal AEC, also from the VIC, will turn the chip on and off by exercising the three-state capability.

The octal latch is an 8-bit register. It is a limited type of register. All it can do is receive the eight bits and hold them. Then, when given an enable signal it outputs them. The latch is a handy device for temporary storage of bits while they are being flashed around the bus lines during the movement of address or data bits. Flip-flop circuits are a lot more versatile than the latch shows them to be.

556 Dual Timer

The 556 chip in the Commodore 128 has two identical sections. There is a flip-flop circuit in each

section that acts as a 1-bit register. The register stores a control bit. One of the control bits turns the reset circuit on and off. The other control bit enables the *NMI pin of the 8502. Both flip-flops are very stable, and once set or reset they will maintain that state unless forced to change. See the Master Schematic.

Control of the control bit is accomplished with special circuits called comparators. A *comparator* circuit acts something like a flip-flop, but it has some differences that make it difficult to use as a register. The main difference being a reference voltage that is applied. A flip-flop does not need this special voltage to store a state.

The comparator circuit in Fig. 11-8 is based around two pnp transistors wired with a common emitter resistor and two collector resistors in parallel to negative $-V_{CC}$. The input signal enters through the base of one pnp. The output signal exits at the collector of the other pnp. The base of the second pnp receives the reference voltage.

The reference signal, as it is applied, forces the second pnp to either cut off or saturate. When the pnp is cutoff, the collector output is the same as

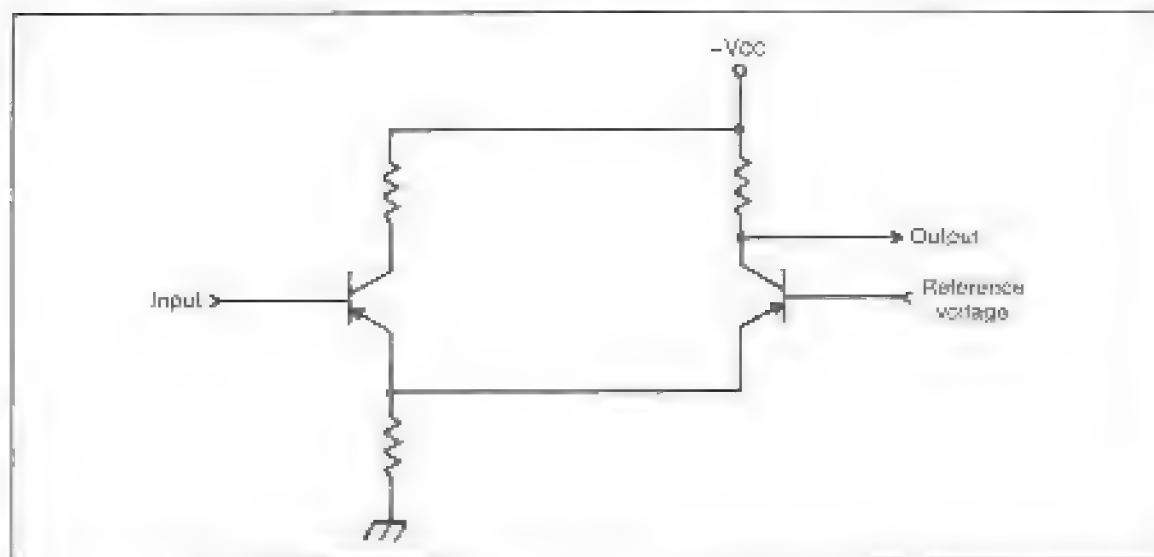


Fig. 11-8 The voltage comparator circuit is often found in computers. It acts somewhat like a flip-flop except that it changes states according to the control of a reference voltage.

-VCC. If the pin is saturating, the collector voltage rises. Therefore, the circuit can attain two stable states.

The input voltage can change the total comparator voltage state if it is changed. If a low is at the input pin, the comparator will exhibit one state. A high at the input changes the comparator state to the other stable state. In effect, the comparator can output a high or a low in response to a high or low at the input.

The output of the comparator controls the state of the flip-flop. For example, in the 556 chip that controls reset, the trigger is tied to +5 volts. When the computer is first turned on, the supply voltage goes from zero volts to +5 volts. This rise to +5 volts triggers the comparator which in turn sets the flip-flop. The output of the flip-flop is a high. The high is changed to a low as it is passed through the NOT gate, a section of the 7406. This low pulse is transferred to the *RES pin of the 8502. This action outputs the starting address of the reset routine. That is how the computer powers up and eventually gets the READY sign and the cursor on the screen.

The other part of the 556 performs a similar job with the *RESTORE signal. When you press Restore on the keyboard, the resultant pulse is sent to the trigger of the 556 section. This produces the output signal *NMI which goes to pin 4 of the 8502 to produce the *NMI effect. Both of these circuits are operating in a one-shot mode.

COMPUTING REGISTERS

When you get right down to it, the digital insides of one computer is quite like any other. The Commodore 128 is different in certain ways, but it computes just like any other brand. Computing is performed in registers. These are the registers in the MPU, memory, and I/O chips. The support chips that we have just discussed to do jobs like latching, buffering, decoding, multiplexing, and other duties to help move the binary highs and lows from place to place in the proper form at the right times. These

same types of support circuits are also found on the main LSI chips to help move the highs and lows from place to place on the chips. The computing itself, though, takes place in the registers of the large, important chips.

What computing boils down to is the manipulation of numbers. All the calculating, all the communications work, the composition of graphics, and the other tasks that the computer is able to do is nothing more than the manipulation of numbers. The numbers are clever codes of the letters and characters but they are still numbers. The numbers are represented in the registers as highs and lows, but the logic states are numbers nevertheless.

The registers can perform a limited group of manipulations on the highs and lows. You have already seen how a register of four bits can add bit after bit, from a start of L-L-L-L to a total of H-H-H-H. This is one of the jobs a register can do. The other jobs are equally easy to comprehend. The registers are not capable of doing any truly complex things. To itemize, registers can add or subtract, shift or rotate, increment or decrement, clear, complement, AND, OR, or XOR.

Besides being able to perform these jobs in the registers, they are built to be able to move the contents of the registers from one register to another. The contents of MPU registers can be stored in RAM registers; the contents of RAM or ROM registers can be loaded back to the MPU and data contents of the 8502 registers can be stored, loaded or swapped among each other. In addition, 8502 register contents can be sent to I/O chips for transfer to peripherals, or peripheral contents can be sent to the 8502 registers via the I/O chips. During servicing it is best to keep in mind that the numbers being manipulated or moved are voltage highs and lows.

During the manipulating and moving of the numbers, the question comes up, where should this data be moved to? The register that answers this question is the program counter. It is a special register that does the addressing. It is examined in detail in Chapter 12.

Shifting

An important ability of a register is shifting. What is shifting? It is the talent a register has of being able to shift bits from side to side. For example, suppose an eight bit register you are working with contains LLHL LHLH. If you instruct the register to "shift the contents one bit to the left," the register will go to a state of LHLL HLHL. The contents of the register moved one bit to the left. If you now told the register to move one bit to the right, the register would then change back to its original state, LLHL LHLH. That is shifting. Another form of shifting is called rotating. Shift and Rotate are important instructions that the MPU and other chips with shift registers can respond to.

Shift registers are not new. A shift register is formed by adding some storage flip-flops to an ordinary register as shown in Fig. 11-9. If an 8-bit register is to be able to shift, it needs seven additional flip-flops. The seven FFs are installed between the eight individual register bits. They will latch the individual bits as they are transferred during the shift. For example, the register contents from above LLHL LHLH, could be shifted one bit to the left in the following way.

First a copy of each H bit is placed into the latch to the left of the register bit. Next, all the register bits are replaced with Ls. The last step is to move the Ls out of the latching FFs into the next bit to the left. The one bit shift left is complete.

The shifting and rotating of bits in a register is a valuable talent. The 8502 responds to machine-

language instructions such as Shift Left, Rotate Right, etc. Upon receipt of such an instruction the register performs the shift. Between the register bits and the latches installed between bits, the highs and lows are shifted. What does the shift do numberwise?

Let's think of the Ls and Hs as binary 0s and 1s. Suppose the shift register is filled with 0000 0001. In decimal, this is the number 1. The register is then given the instruction "shift one bit to the left." It does so and the register then reads 0000 0010. The 1 moved to the left and the value of the register in decimal is now 2. You are probably thinking, why this is exactly like the binary adder. No it's not. Even though the adder, after an addition of 1, totaled 2, this shift register has not added but doubled its decimal value.

This becomes clearer as we instruct the register to shift another bit to the left. The register then shows 0000 0100. This is the decimal value of 4. The register has doubled again. The adder register, after another add would be three. Should we shift left once more the register goes to 0000 1000, which is decimal 8. The shift doubles the value of the register when it goes one bit to the left. This fact becomes very valuable when you are manipulating numbers as you compute.

If the shift left produces a doubling of numbers, what happens to the numbers during a shift right? The value of the register is divided in half as the register is made to shift right. One move to a higher significant bit multiplies the register value by two.

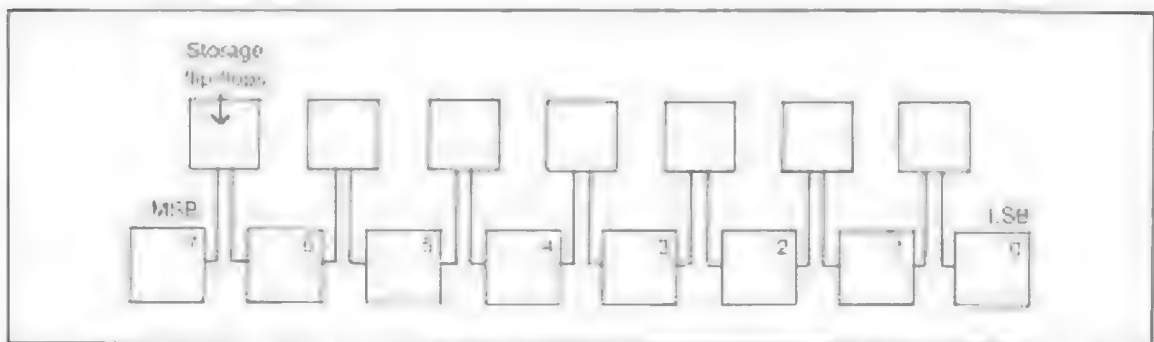


Fig. 11-9 The shift register needs additional storage flip-flops between register bits. First, the states that are in the bit holders are transferred to the storage flip-flops. Next the main register bits are all cleared. Lastly, the stored bits are returned to the main register but shifted one bit to the right or left, whichever is desired.

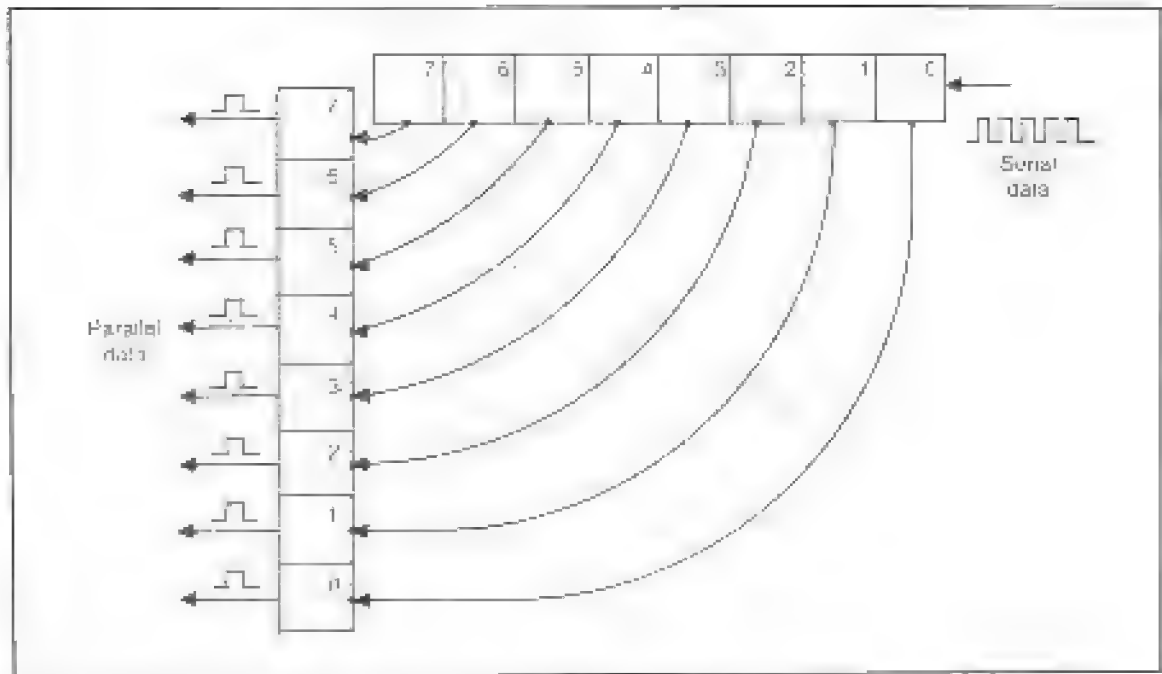


Fig. 11-10 The shift register is handy to convert serial data to parallel, or vice versa.

One move to a lower significant value divides the register value by 2. This is one important function of a register during the task of number crunching.

The shift register has other jobs it can do. It aids during the transferring of numbers from place to place. As data is transferred from register to register it is coded, decoded, latched, buffered, multiplexed, and changed in numerous other ways to get the data to the right place, in the right form, at precisely the correct time. Sometimes a byte of data is transferred in parallel fashion, all bits moving over a byte-sized bus, eight abreast. Other times the byte is transferred in a serial way, over one wire, one bit at a time, in a long single file column. There also comes a time when the parallel group of bits must leave the eight lines to the bus and enter one wire in single file.

Sometimes a serial signal must be converted to enter a parallel bus line. The shift register can do the honors. This use of the shift register is strictly a transfer medium and has no bearing on the numerical value of the bits. There are a number of these types of shift registers in the I/O circuits of the

CL28. The shift register can receive a serial data input at its lowest significant bit. The bits enter one at a time and are immediately shifted to higher bits. As a full byte of bits enters the byte sized register, the bits fill the register. The serial bits are thus latched into the register. Each bit holder has an output terminal. The outputs are all connected to another latch. This latch is connected to the parallel lines of the data bus. The latch is opened and the bits flow onto the data bus as a parallel signal. Figure 11-10 illustrates this process.

The reverse is also part of the shift register's duties. The parallel data bus is able to latch its contents. The latch is then connected to the shift register. The latch empties into the shift register in a parallel fashion. The shift register can then shift the bits to the right and output them one at a time out of the least significant bit.

Clearing

An important job that registers can do is CLEAR. The word clear means to reset a register or bit to a low, or numerically a 0. Figure 11-11 il-

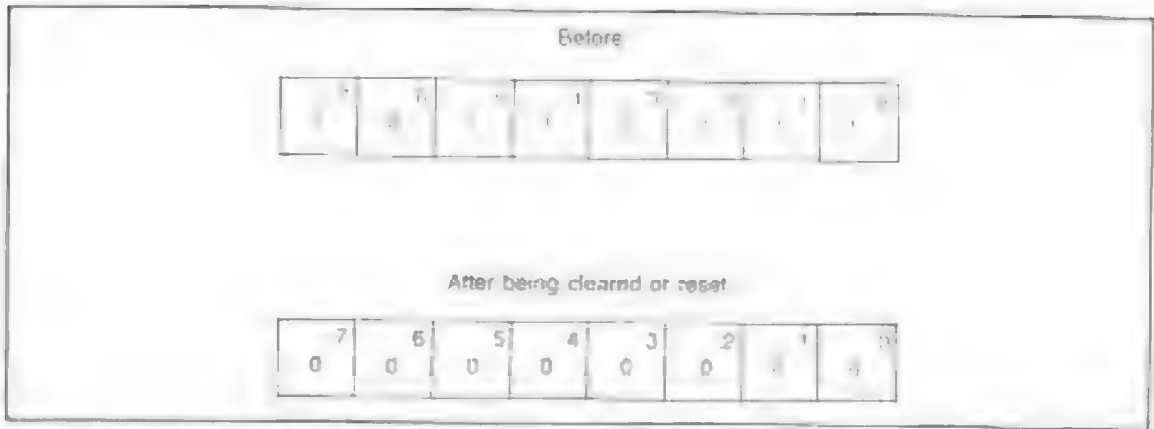


Fig. 11-11 When a register is instructed to clear or reset, all the bits are replaced with 0s, no matter what was there beforehand.

illustrates this idea. A register usually must start operating in a clear state of 0.

While the various names for this process all mean the same thing, there are some fine distinctions. The word reset seems awkward when referring to the start of a register. It seems to make more sense if a register is clear when it starts off and attains a state of reset the second time it goes to a 0. A register should become set to a 1 before it can then be reset to a 0.

Whatever the variations in definitions, you can think of the register being clear, as long as it is in a state of 0. When a register is cleared all the bits are reset to 0. There are many electronic ways a

register can be zeroed. You can add the correct states to a register and end up with all 0s. The register can be shifted into a complete 0 state. The register can have 0s POKEd into it. The register can have AND or OR logic applied to it to produce 0s. The clearing of registers is an important ability and besides programming is used extensively in the factory during quality control testing.

Complementing

When a register is complemented, all the highs are changed to lows and the lows are changed to highs. This process is shown in Fig. 11-12. This is easy to do electronically. If you transfer a byte of

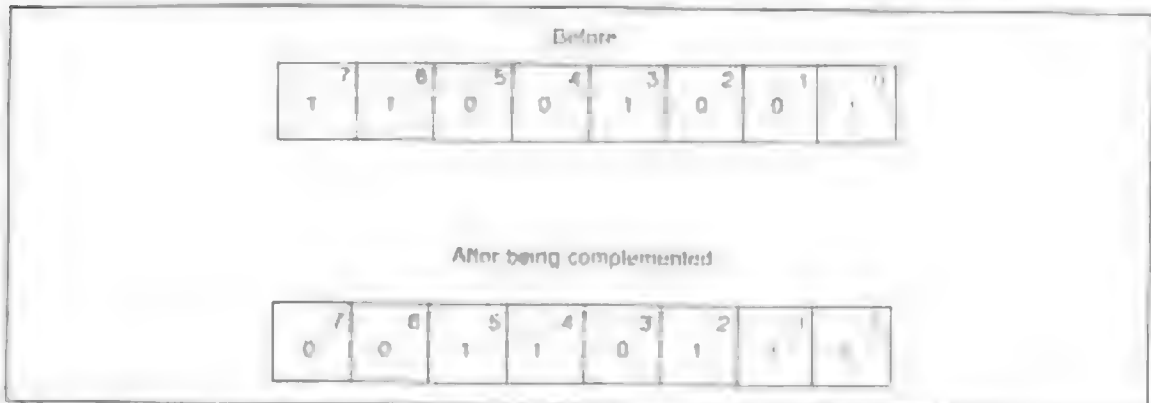


Fig. 11-12 If a register is instructed to complement, all 0s are changed to 1s and all 1s to 0s.

bits from one register to another you can change the state of the individual bits. All you need do is pass each bit through a NOT gate as it moves from register to register. The 1s will change to 0 and the 0s will become 1s.

At first glance, this looks like an interesting but not very useful data manipulation. What value can the complement maneuver be? As it works out, in binary arithmetic complementing is an important function.

The circuits needed to add binary numbers are relatively simple. We've explored the binary counter circuit somewhat. With that circuit it is easy to add numbers. It is also easy and straightforward to add one register to another. Simple multiplication can also be performed since multiplication is only the addition of a lot of the same numbers. The lightning fast registers perform multiplication by simply adding the group of same numbers together. It doesn't need a multiplication table like a human does. To get 4×8 , the registers just add $4 + 4 + 4 + 4 + 4 + 4$.

While addition and multiplication lend themselves to FF registers, subtraction does not. In order to design circuits that will subtract quickly, expensive and difficult designs have to be manufactured. There is an easier way to get the computer to subtract. It is called *subtraction by addition of the complements*. This is tricky, but once understood, simple.

It works out, and you'll have to take my word for it, that if you take the complement of a binary number, and add it in a certain way to another binary number, the result will be the same as if you had subtracted the original number. That is how the 8502 in the C128 does subtraction.

I'm not going into a proof; there are plenty of books around on the subject if you are curious. The knowledge of the subtraction method is not vital to the repair of the C128. For your information though, the C128 performs the subtraction with the two's complement.

When you change the 1s to 0s and the 0s to 1s in a register, it is called one's complement. In order to obtain the two's complement, you add a 1

to the register contents. That makes the register contain the two's complement. In order to subtract the original register number, the two's complement number is added. The result of this addition is the same as subtracting the original numbers.

In case you are curious about complex multiplication and division of numbers, microcomputers usually have special subroutines installed on their ROM chips. These routines are called every time a long multiplication or division problem has to be solved. This has nothing to do with complementing. The important use of complementing is subtraction. There are other things a programmer might do. These are of little interest during troubleshooting or repair.

Increment, Decrement, and Jump

The Commodore 128 has a number of registers that are continually counting up, counting down, or just jumping from one binary number to another. For example, the 16-bit Program Counter is constantly doing this. The PC always contains a binary number. The number is the next address in the memory map that will be contacted. When the number advances by one, it is said to be *incremented* by one. If the number is reduced by one, it is *decremented* by one. Should the number just change to a number that is far from the previous number, it is making a *jump*.

These three register abilities are vital to the operation of the PC. The PC is attached to the 16 address bus lines. The 8502 gets connected to whatever number the PC puts on the address bus.

The PC is built to automatically increment by one after every address cycle while running a program. It does the incrementing continually unless it is instructed by the program to do otherwise. The incrementing is simply adding a 1 to the least significant bit of the register. If there is a low in the LSB it is replaced by a high. Should there be a high in the LSB, the high is transferred to the next higher bit and the LSB ends up with a low. Refer to Fig. 11-13. As each new high is applied to the LSB the total binary number is incremented by 1. Note that this process is addition in the register and not shift-

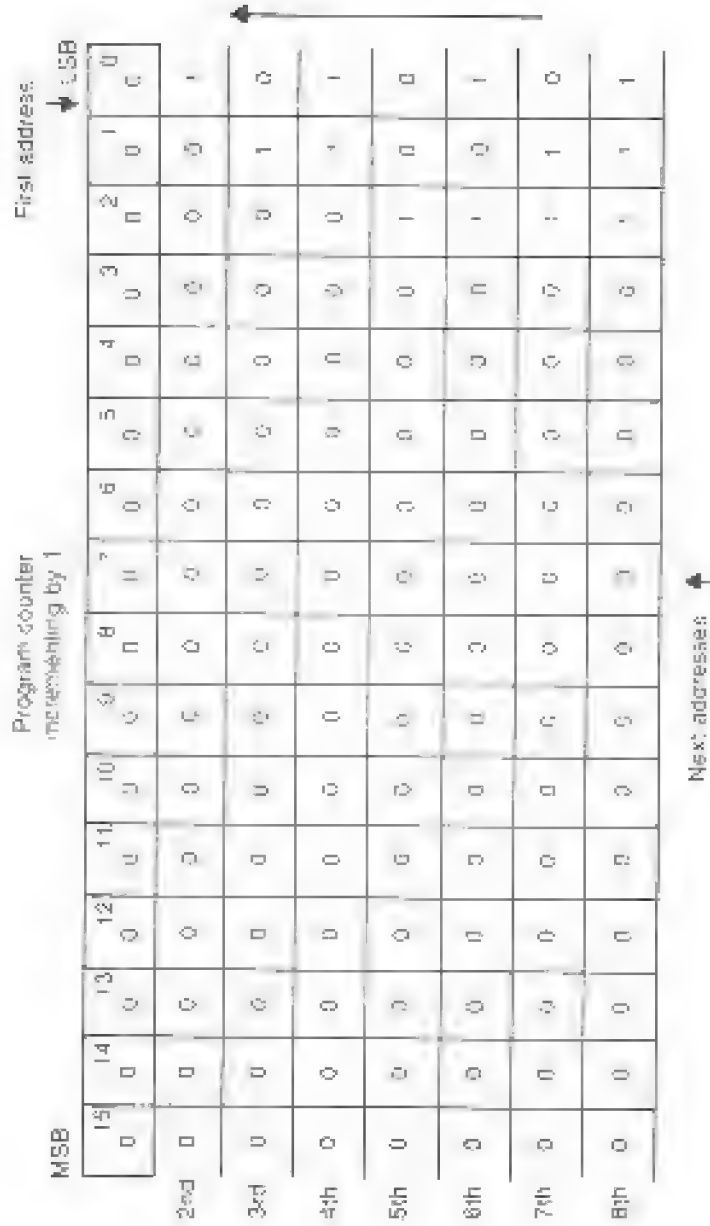


Fig. 1-3. A register like the program counter is able to increment automatically. Other registers need increment or decrement instructions to affect the change.

ANDing two registers

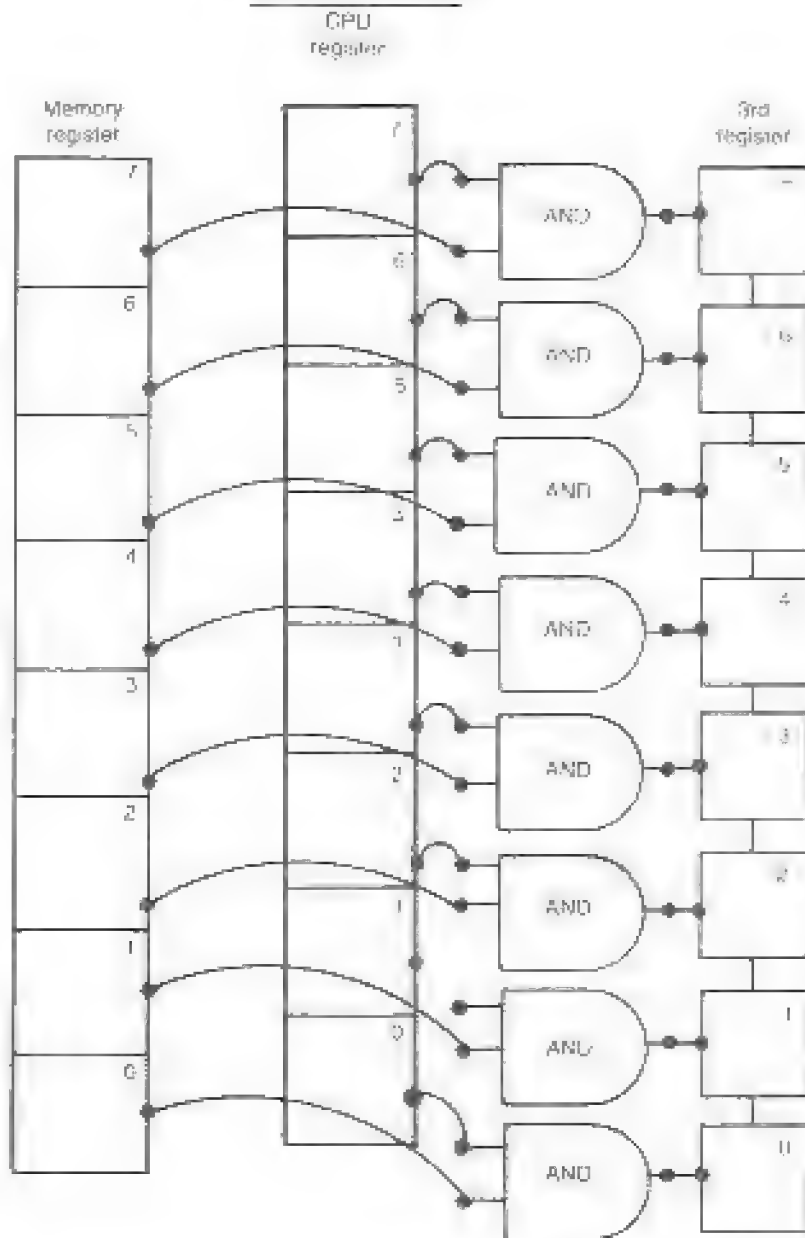


Fig. 11-14 When a complete register is ANDed with a second register bit by bit, the output results are stored in a third register.

ing, which continually doubles the total with each shift. The numbers increase one at a time as the addition carries from bit to higher bit.

Decrementing is the reverse of incrementing. During the decrement job, the total binary number in a register is reduced by 1. For instance, in the 8502 there are two registers called the X and Y index registers. One of the techniques of indexing has to do with decrementing a register after each addressing cycle. As a 1 is removed the entire register binary number is reduced by 1.

Incrementing and decrementing registers by 1 is a common practice during computing. In addition, registers can be changed drastically. For example, the program counter can receive a JUMP instruction followed by a large number. When the JUMP arrives, the number could be added to the present address on the PC. A new larger number is then produced on the PC. The new number is placed on the address bus. The addressing then jumps over a group of numbers and lands on the new number placed on the bus. This is a very important function of the program counter register.

ANDing and ORing

AND and OR gates have been discussed. With the aid of AND or OR gates, registers can also per-

form the AND and OR jobs. As Fig. 11-14 shows, if you connect the outputs of a byte sized memory location and an 8502 register to eight AND gates, the outputs of the two registers can be ANDed together. The outputs of the AND gates could then be placed into a third register for storage. The third register would contain the results of the ANDed registers.

The ORing of two registers can be accomplished in the same way. The same position bits from two registers are injected into OR gates and the outputs of the gates placed into a third register. XORing two registers can also be easily performed with this technique. ANDing, ORing, and XORing are all easily done between memory locations and the accumulator register in the 8502. The XORing is called EOR by Commodore.

There will be a lot more about registers in the following discussions of the details on the main LSI chips. However, you'll find that there isn't really too much variety in what FF registers are doing. They only do a few jobs. They all can store, latch, and pass on binary data. Then some of them, but not all, are able to increment, decrement, clear, complement, shift, and rotate. Lastly, a chosen few are able to add, subtract, multiply, divide, AND, OR, and XOR. That's about all registers can do. That's all they have to do to get the computing done.

12. 8502 Microprocessor

The 8502 is the main actor in the drama of C128 and C64 mode computing. The Z80 is its counterpart for CP/M mode. The 8502, when it fails, can produce the following symptoms. First of all, it can cause a complete breakdown with these conditions. The pilot light will turn on okay, power will be present in the computer at all +5 volt connections, yet nothing will be happening. Failure of the 8502, under these conditions, is indicated. The 8502 is the number one suspect and should be tested.

The 8502 could possibly also be the culprit if one of the modes is gone but the other one is still okay. However, the 8502 is not the prime suspect in these cases. In fact, it is probably good when only one mode fails.

The prime suspects when the C128 mode conks out and the C64 mode is fine, are the MMU, U7, the PLA, U11 and the ROMs which are operating the C128 mode. They are U33 and U34, ROMs 2 and 3. On the other hand, if the C128 mode is down and the C64 mode is okay, then the suspects be-

come the U7 MMU, U11 PLA and ROM1. ROM1 contains the C64 operating system and the MMU and PLA are involved because they perform the selecting and addressing of the ROMs. It is possible that the 8502 could have some odd trouble that is producing the symptoms, but is it probably alright.

The Z80, while not involved in the operation of the C128 and C64 modes, is also a suspect. It so happens that the Z80 is the chip that gets the C128 computer started. The Z80, covered in the next chapter, is given control when you first turn on the machine. Once the machine is running, the Z80 then turns control over to the 8502 and its workmates. Therefore the possibility always exists that a bad Z80 is killing the C128 and C64 modes because it is not starting up the machine. When the C128 does not come on properly, then line up the symptoms and puzzle out what could be happening according to the existing symptoms. The immediate suspects are the 8502, Z80, MMU, PLA, ROM1, ROM2 and ROM3.

Note that the symptom specifies that power is

being supplied properly. If there is no power, this chapter does not cover the trouble. Chapter 24 has the no-power story.

ADDRESS AND DATA BUS LINES

When the modes are gone and power is okay, then the first troubleshooting step is to test the address and data bus lines at the 8502 pins. Coming out of the pins in Fig. 12-1 are the address lines: A15-A0. When these lines are operating okay, then they are running continually and outputting various sets of bits according to the data that is coming from the operating system in the ROMs. According to instructions, the address lines are opening up locations and either reading the location's data or writing to the locations. The reading and writing cause bits to be constantly traveling the data bus.

You can detect the bit movement on the 16 address lines and eight data lines with a logic probe. The bits, both address and data bits, travel as waves or pulses. When the bits are in motion, the pulsing is called *activity*. The logic probe will turn on its PULSE light.

Inside the 8502, by each pin connection, is a buffer. The 16 address buffers are two sets of eight buffers each. There are eight called address bus high and eight called address bus low. The eight high pins connect to lines A15-A8 and the low pins are to lines A7-A0. These designations follow true to form back to the innards of the chip. Deep inside are the two sections of the program counter: program counter high and program counter low.

Also deep in the chip are the other registers. Figure 12-2 shows how these registers interact. The stack pointer register connects to the low part of the addressing system. The ALU connects to both the low and high sections. The input data latch also connects to both the low and high.

When you touch down on the addressing pins with the logic probe, the address bits are coming from one of these four registers. The most prominent register is the 16-bit program counter. It can address one of the 64K memory banks because it has 16 pins to form bit combinations

The ALU and the data latch also can address the entire memory map with their outputs. The stack pointer, though, is only attached to the lower address lines, A7-A0. It can only point to 256 memory locations with its eight-bit capacity.

In the RAM locations, a special section is assigned to be the stack. The stack is given 256 addresses, from decimal location 256 through 511, as in Fig. 12-3. The 8502 uses this area to store temporary data, subroutine control bits and interrupt bits. I give more detail on the stack later in this chapter.

The eight data bus lines, unlike the one-way outgoing address lines, can both input and output data bits. When data exits the D7-D0 pins, the bits are coming out of a set of eight data bus buffers. The bits continue on to whatever location had just been addressed. When the bits arrive at the location they enter registers and are stored or produce some sort of action. It is said that the processor has just executed a write.

When the processor is doing a read, though, the bits are entering the D7-D0 pins. Once inside the chip, the bits have two possible places they can go. If the bits are ordinary data, they go to the same data bus buffer stage that also can output bits. Two sets of eight buffers are in the stage. One set is pointed out and handles the bits that are leaving the processor. The other eight buffers are pointed inward and handle the input bits. The input bits are buffered and enter the processor circuits.

The other place incoming bits can go to is the *instruction register*, known as the IR. When a program is being read by the processor, first an instruction is read and then the data the instruction refers to. The instructions go to the IR and the data goes to the data bus buffer. Figure 12-4 shows the arrangement.

STACK

There are a number of important registers in the 8502 that will be covered now. One such register is the stack pointer. The word pointer simply refers to an address. The stack is allocated dec-

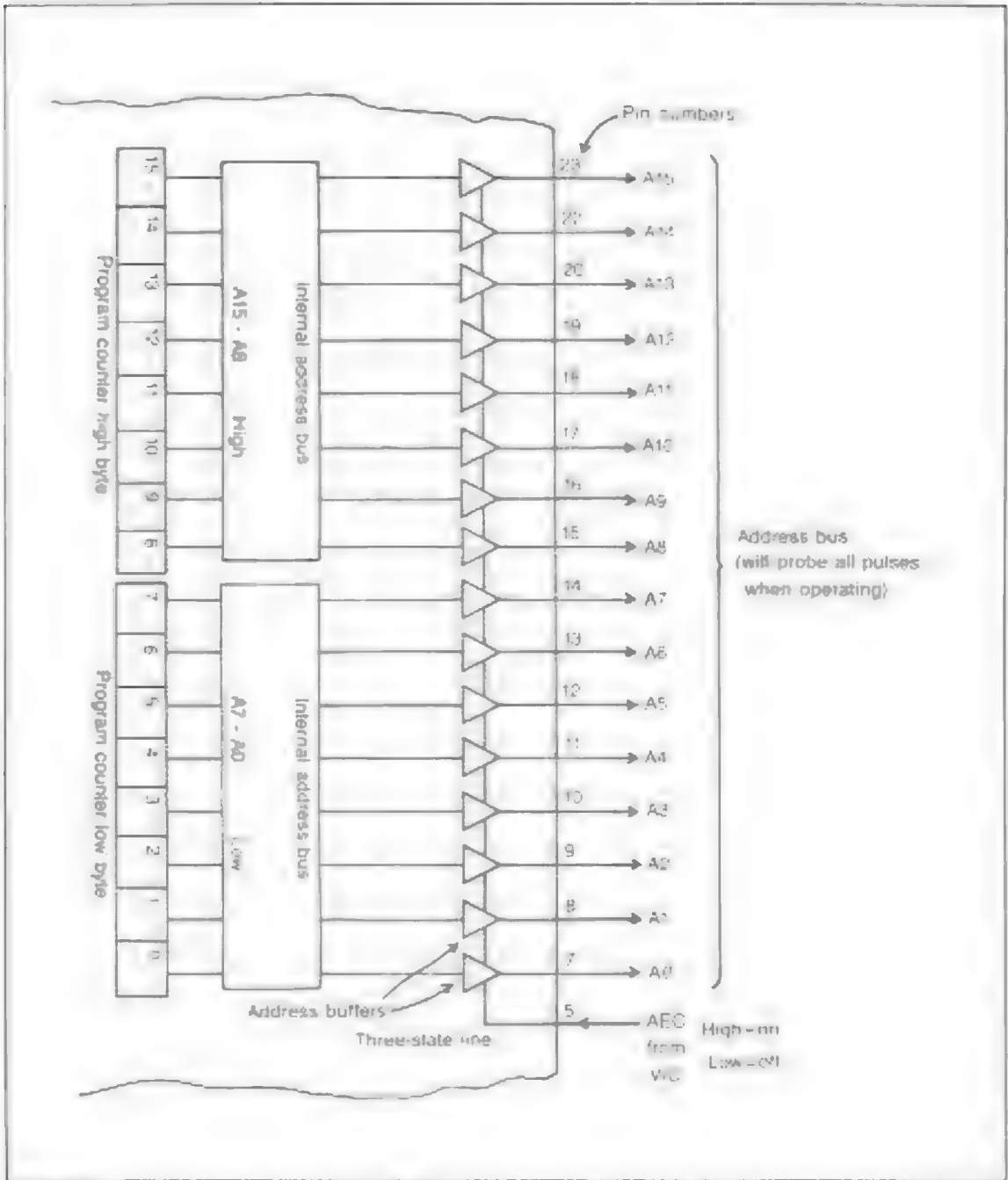


Fig. 12-1 The 8502 has a set of three state buffers in its address output circuit that can be turned off and on with an AEC signal from VIC.

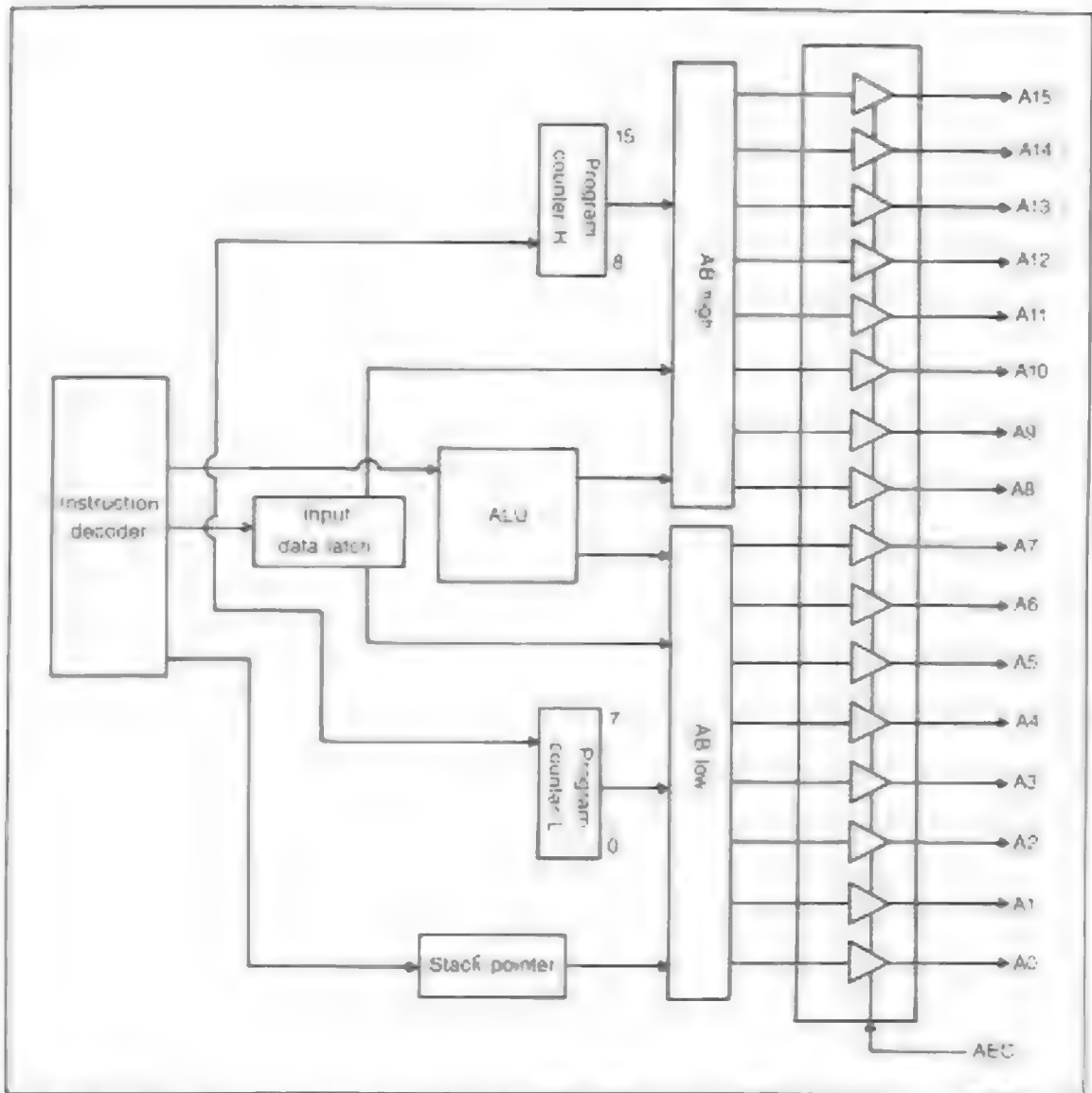


Fig. 12-2 The stack pointer is able to form addresses from 256 to 511 only. It is connected only to the lower address data, A0-A7, of the program counter.

final addresses 256-511. The stack pointer is nine bits wide (see Fig. 12-3). Nine bits are able to form enough combinations to hold a stack address. The addresses that the stack pointer register are able to hold are, in binary: 1 0000 0000 to 1 1111 1111. The stack starts with 1 1111 1111 which is 511 in decimal.

1 1111 1111 is the address when the stack is empty. If the 8502 PUSHes a byte into the stack, then the byte enters 1 1111 1111 and the pointer decrements to the next lower empty number: 1 1111 1110. The pointer register keeps decrementing as the stack fills up. The stack is able to store a page (256) of bytes.

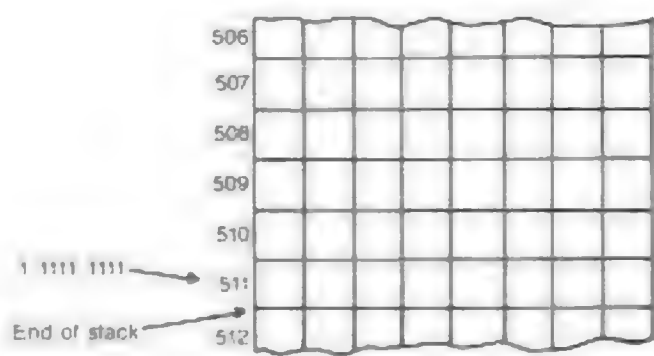
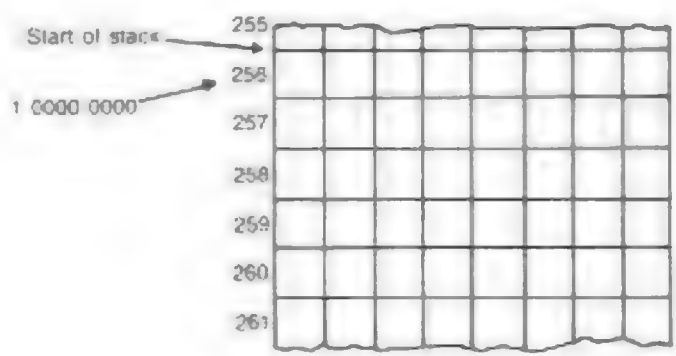
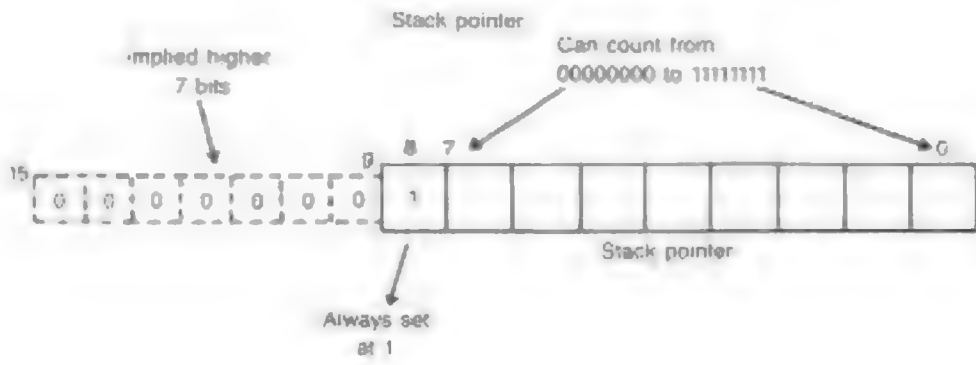


Fig. 12-3. The stack pointer register is actually nine bits wide. The ninth bit (numbered 8 since the first bit is numbered 0), is always set at 1. This gives the register the range of 1 0000 0000 to 1 1111 1111, which is 256-511 in decimal.

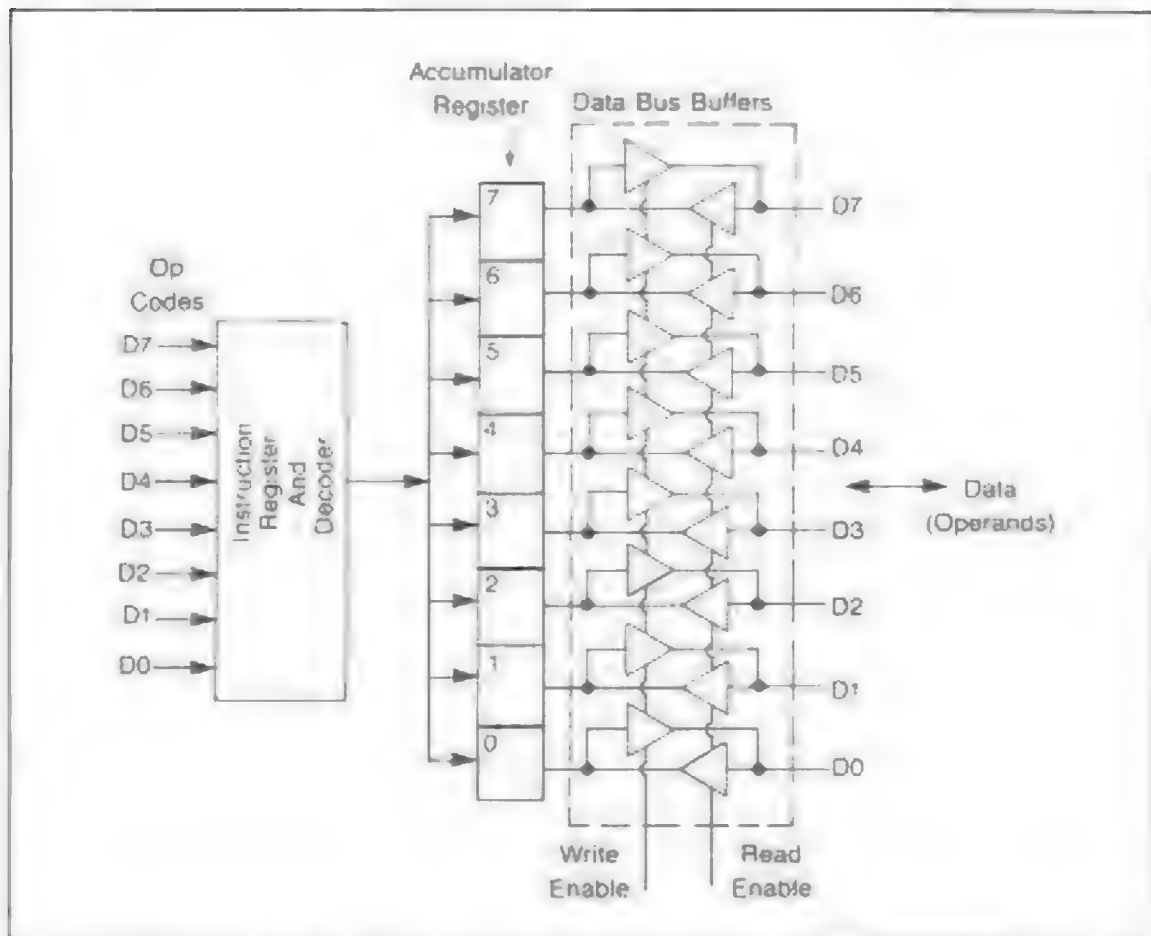


Fig. 12-4. The accumulator register receives instruction bits from the instruction register and decoder. The accumulator also receives and outputs data bits to and from the data bus. The data bus buffers control the direction in which data bits travel.

If the stack is near full, then the pointer could read 1 0000 0100. Should the 8502 PULL a byte off the stack, the pointer would then increment and point to the next empty byte holder: 1 0000 0101. The stack area in RAM is the repository where the states of the registers of the 8502 can be stored, when need be. For example, as programs are being run there are usually lots of interrupts. The interrupt could be a peripheral device that needs immediate attention from the 8502. A device that will have priority over the running program.

The interrupt forces the 8502 to stop at the next convenient moment. Meanwhile, all the registers in the 8502 are in the midst of a lot of data manipula-

tion. If the 8502 should just stop to service the peripheral interrupt, what happens to all the data in the registers? It could all be lost as the 8502 services the interrupt.

As you may have guessed, all the data in the pertinent registers are PUSHED onto the stack before the 8502 deals with the interrupt. The register bytes are stored in the stack temporarily. First the contents of the 16-bit program counter are pushed onto two stack locations. Then the contents of the X and Y index registers are pushed on two more locations. Following that, the one byte in the accumulator is pushed on. Lastly the eight bits in the condition code register are placed into one byte.

Once the registers are stacked, the 8502 then services the interrupt.

At the end of the interrupt routine, the 8502 is ready to resume the program that was in progress when the interrupt arrived. The 8502 now must restore that registers to the state in which they had been. The 8502 starts pulling or as it is called *pop ping*, the bytes out of the stack. The pointer is addressing the last byte to go in the condition code register. The state of the CC, which was the last byte (Last In First Out, LIFO) is popped and returned to its register in the MPU. Then the rest of the bytes are popped in the reverse order that they entered. The pointer ends up at the original location it had been addressing.

The bits in the stack pointer are not stored in the stack area because they are not affected by the interrupt. They remain in the 8502 to remember where the bytes were stacked during the interrupt.

It is said that the stack is found in "page 1" of the 64K RAM memory bank. A 64K bank is par-

tioned off into "pages." A *page* is simply 256 bytes. Page 0 holds decimal locations 0-255. Page 1 has locations 256-511. In 64K there is enough room for 256 pages. The last page in the 64K bank is 255.

ARITHMETIC LOGIC UNIT

The 8502 is quite like the 6502 that is in the VIC 20 and the 6510 found in the C64. All of these MPUs are the result of more than 30 years of evolution. There is nothing dramatically new about the microprocessor except for the fact that their circuits are microscopic. Essentially the same electronic manipulations have been going on for years in vacuum tubes, transistors and then integrated circuits. The ALU, the *arithmetic logic unit*, is not a new innovation. It is as old as computers. It is also the most vital section of a computing system. It does the data manipulation.

A typical ALU has two byte-sized inputs and one eight-bit output, as Fig. 12-5 shows. The ALU is

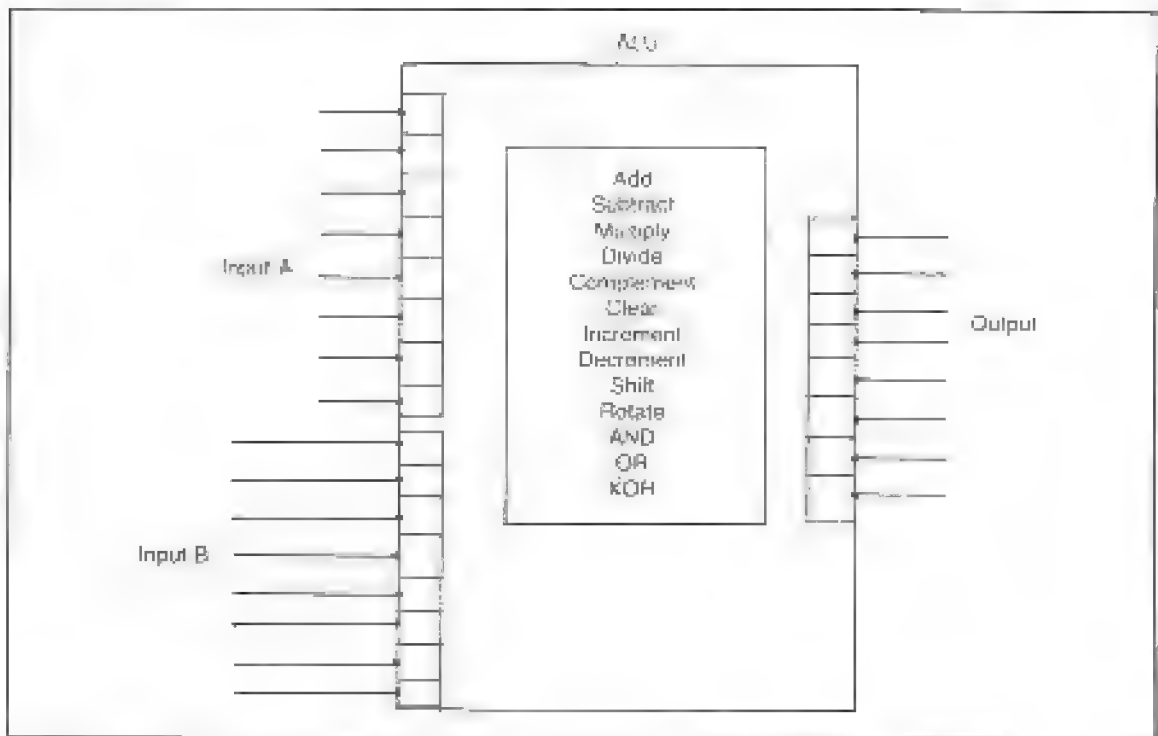


Fig. 12-5. The ALU has two byte-sized inputs and an eight-bit output.

able to do the following things as it crunches the two inputs to a single output. First it can add the two inputs into a single output result. The addition, with some mathematical trickery can also result in subtraction, multiplication and division. Next the ALU can *complement* its internal registers, making all 1's into 0's and all 0's into 1's. It can *clear* one of its inputs; that is, change the inputs to 0. It can become a shift register and shift all the bits in one of the inputs either to the right or to the left. Finally it can perform logic manipulations such as AND, OR and Exclusive OR.

Shifting

When a *shift left* is made, the value of a register contents is doubled. If a *shift right* is made, the value in the register is divided by two. When the shift right is made, the lowest significant bit falls out of the register into a special bit holder in the condition code register and a 0 is forced into the most significant bit. During a shift left the MS bit falls out into the bit holder and a 0 is placed into the LS bit. Figure 12-6 depicts the way in which the shift works.

The shift ability has a variation called *rotate*. Rotating works in the same fashion as shifting except for one major difference: when you shift a byte, one of the end bits falls into the bit holder in the CC register.

With the rotate function, the bit that falls out does not get lost, as shown in Fig. 12-7. It is caught in the CC bit holder like the shift function. However,

the contents of the bit holder are brought around to the other end of the register and installed there. The same thing happens with either a rotate right or rotate left, the bit just travels in the other direction. The CC bit holder acts as a ninth bit of the shift register that connects the two ends of the register into a loop. Incidentally, the CC register is also known as a *flag register* and is covered later in this chapter.

Logic Manipulations

As the name implies the ALU, besides performing arithmetic calculations, also does logic manipulations. Logic is a form of mathematics that deals with the NOT, AND, OR, etc. calculations. In the ALU are forms of these gates to do the logic work. For instance, the *complementing* mentioned earlier is actually a logic job. When a register is *complemented* it has all its bits run through NOT gates which change 1's to 0's and 0's to 1's.

When the ALU ANDs two inputs, it uses one AND gate for every bit position, 7-0, in the two inputs. The two MS positions, bits 7, can be the input to the position 7 AND gate. The resultant AND output becomes the output bit 7. Each set of bits can be ANDed in the same way. An AND operation proceeds simultaneously in parallel. The operation forms a total ANDed byte.

When you AND a byte with another byte, all the bit sets that had 0's will output a 0. All the bit sets that had two 1's will output a 1. ANDing a byte

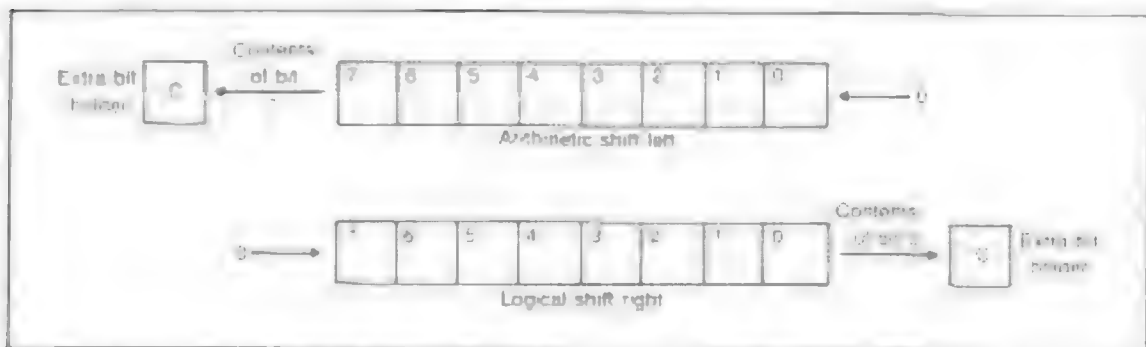


Fig. 12-6. During an Arithmetic Shift Left instruction, a 0 is pushed into bit 0, the contents of the register are all shifted one bit to the left; and the contents of bit 7 fall into the \overline{C} flag of the status register. During a Logical Shift Right, a 0 is pushed into bit 7, the register contents are shifted one bit right and the contents of bit 0 fall into the \overline{C} flag.

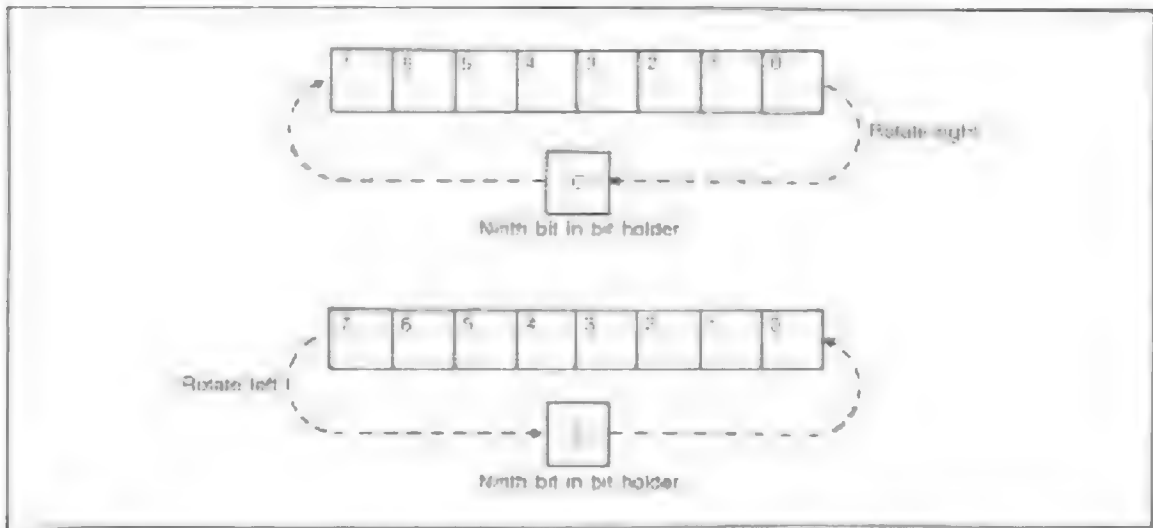


Fig. 12-7 During Rotate instructions, the loose bit falls into the C flag, but the contents of the C flag are pushed out and into the bit that becomes empty due to the instruction. The C flag connects the register into a closed ring and acts as a ninth bit.

allows you to mask a byte. If you want to make sure that certain bit positions will end up with 0's, then you AND those bits with 0's. This procedure, for machine language programmers, shown in Fig. 12-8, is known as masking off certain bit positions.

On the other hand if you want bit positions to remain unchanged, then they are to be ANDed with 1's. These bits are thus not masked and will be the same before and after ANDing.

The ALU ORs two input bytes in the same way. Whenever one of two inputs into an OR gate is a 1, the output is a 1. The only way two ORed bits can produce a 0 is if both inputs are 0's. The total result of one byte being ORed with another byte is an ORed output byte.

When it is necessary to change certain bits in one of the ALU inputs to 1's, all you have to do is OR the chosen bits with 1's. Figure 12-9 demonstrates the process. Whether the bit is a 0 or a 1, when it is ORed with a 1 it will be a 1 output. The bits that are ORed with a 0 will not have any changes. The 1's will remain 1's and the 0's will remain 0's.

The 8502 is also able to Exclusive OR two input bytes. When two input bits are the same, either a pair of 1's or 0's, then the output bit will be

a 0. If the two bits are different, a 1 and a 0, in either way, then the output bit will be 1. The total result of a byte being EORed with another byte is a total EORed byte. The EOR operation is used to detect errors and correct code. While this is mostly the programmer's province, these logical manipulations can be installed in engineering diagnostic programs to detect subtle circuit faults.

ACCUMULATOR

Veteran computer people refer to the accumulator as a *scratch pad*. This name came about because it is a holding place for the binary numbers that go in and out of the computer. The accumulator register in the 8502 is an eight-bit place wired between the data bus, D7-D0, and the ALU. The programmer thinks of the accumulator but doesn't concern himself with the ALU. As far as the programmer is concerned, the ALU is just a calculator that the accumulator uses.

The accumulator is the most prominent register. It performs all of the arithmetic and logic manipulations with the aid of the ALU. The accumulator is wired, with eight data lines, to the internal data bus of the 8502. The internal data bus leads to the data bus buffers and then out to pins D7-D0.

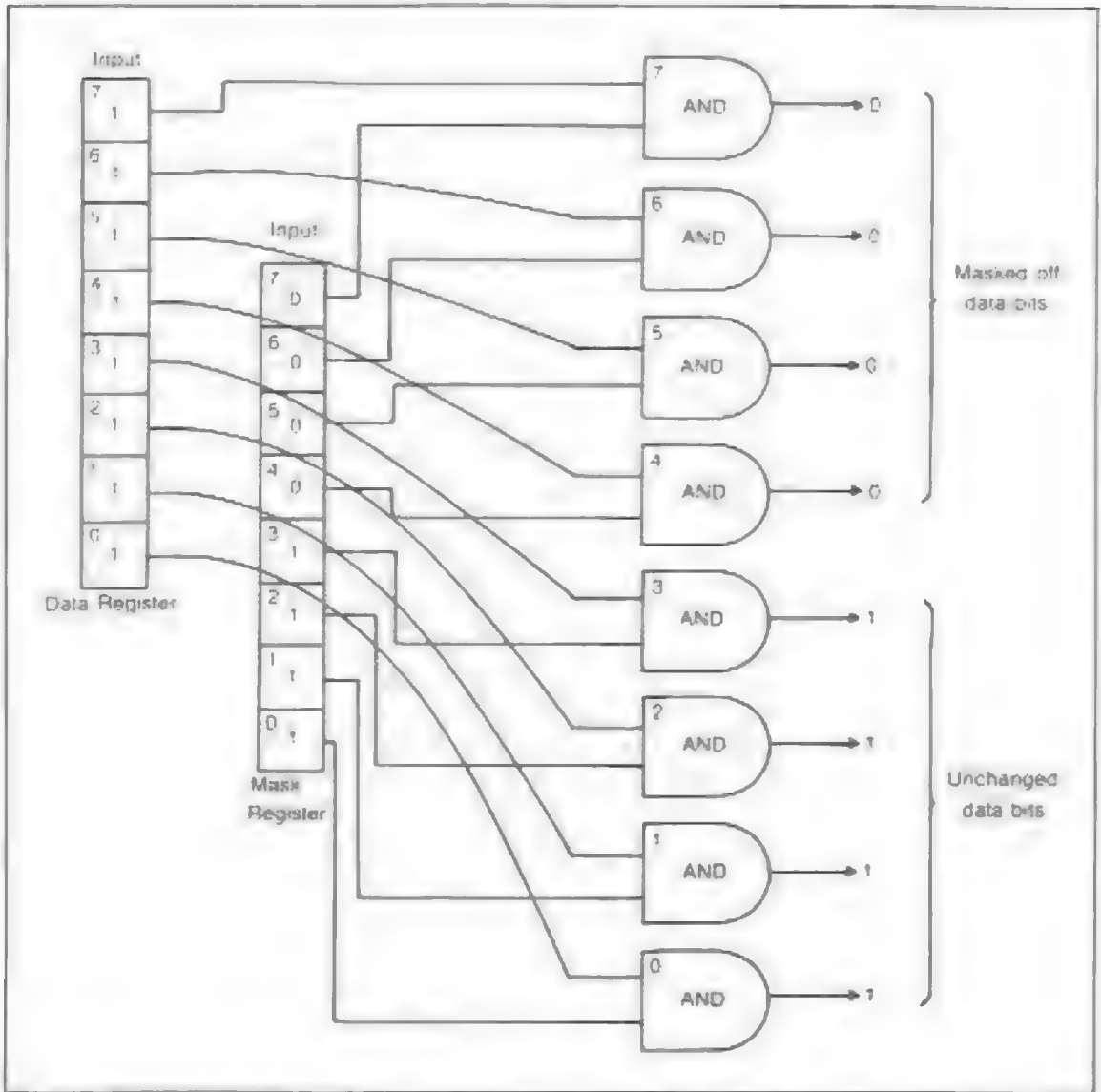


Fig. 12-8 Register bits can be masked off to 0's by ANDing them with 0's. Bits ANDed with 1's will remain the same.

The connection scheme is shown in Fig. 12-10. When you are probing the eight data bus pins, you are actually examining the eight bits of the accumulator. When the computer is operating okay the logic probe should show pulses on all eight pins just like the address pins show pulses on all 16 pins.

The accumulator can either receive data from the data bus or output data to the bus lines. The

data bus buffers decide which way the traffic should be allowed to flow. Inside the microscopic data bus buffer are 16 YES gates. The gates are three-state. Eight gates are wired to send bits out of the 8502, and the other eight gates are wired to receive bits from the memory map locations. The gate wiring is called "head-to-toe." The 16 gates are all in series with the eight data lines. Two gates to a line, the

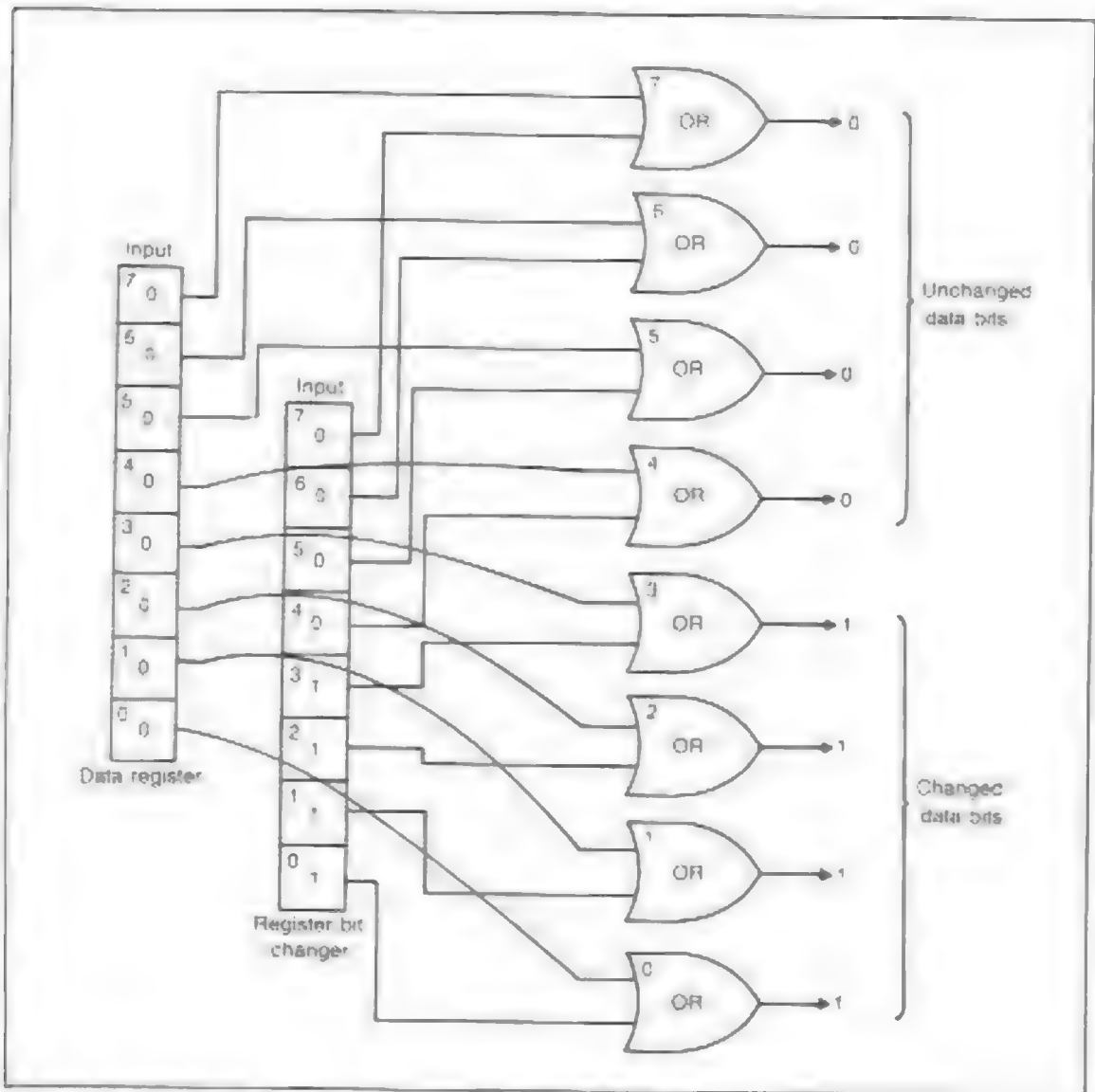


Fig. 12-9 Register bits can be changed to 1's by ORing them with 1's. Those ORed with 0's will remain unchanged.

two gates though are in parallel with each other. The two gates are each pointed in the opposite direction from each other.

The three-state YES gates are controlled by the R/W line. When the line is high, the eight gates that process the read operation are on and the eight gates that allow the write are turned off. If the R/W line goes low, then the reverse takes place. The

eight read gates go off and the write gates turn on. The accumulator accepts data during a read and outputs data if a write is ordered.

INSTRUCTION SET

The 8502 is able to receive instructions in the form of bytes of binary bits. In a byte there can be a possible 256 different combinations of bits. How-

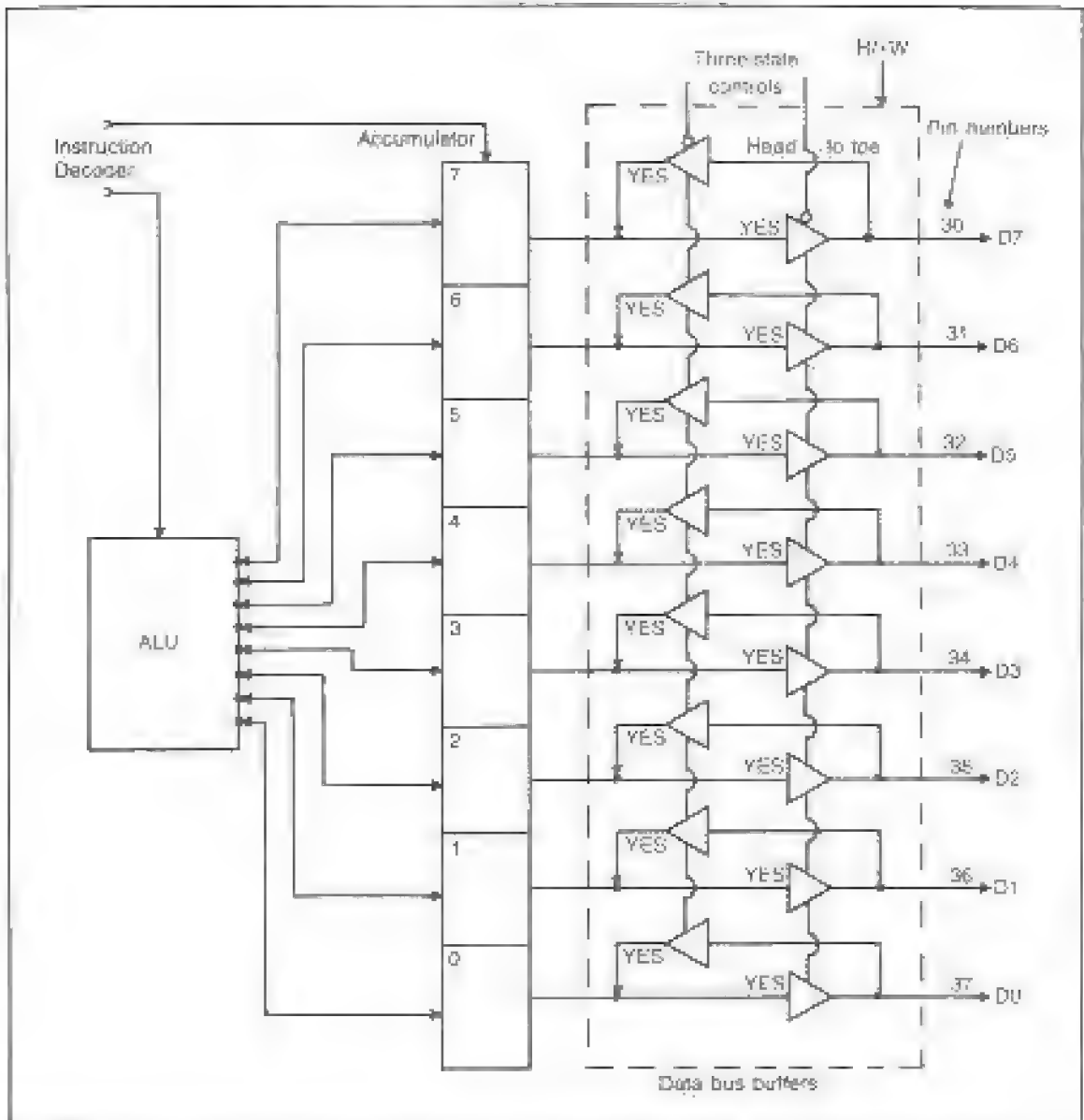


Fig. 12-10. When you are probing the data bus pins, you are actually examining the activity in the accumulator.

even, today's 8502 does not need that many instructions. All it uses is 56. Incidentally, the instruction set of the 8502 is identical to its ancestor, the 6502.

The 8502 programs are run in bytes. When you write a BASIC program, the BASIC ROM changes

the program lines you write into bytes of highs and lows. These bytes are then stored in RAM in sequential addresses. There are other programs stored in ROM. They are also in sequential addresses. The RAM and ROM are able to output

bytes to the 8502. The bytes are what the program consists of and the 8502 processes byte after byte. This is what a processor does. It runs programs.

The 56 bytes that make up the instruction set all have a different combination of highs and lows. The 8502 has internal circuitry that can recognize the 56 different eight-bit sets.

To use an analogy, suppose you have a combination lock that requires eight different numbers to open it up. The lock will remain locked till you enter the numbers in the correct sequence. After you enter the numbers all the tumblers will fall and the lock will open. You could say, the eight combination numbers of the lock are an instruction that says "open." When applied, the lock will always open.

The instruction bytes operate in the same way. When an instruction byte is applied to the 8502 from a memory location where it was stored, it activates one of 56 circuits. Each circuit is built to perform a job when it is activated. The circuit responds by doing the specific job the instruction byte calls for.

Instruction Byte

Some of the bytes in the stored program form instructions. Other bytes in the program are not instructions, they are data. The instructions are specific to the instruction set. The data is the information that the instructions process.

Typical instructions are store, load, add, subtract, complement, clear, increment, shift, AND and OR. There are also instructions that will change addresses, such as jump, branch, call and return.

Each of the bits in an instruction, shown in Fig. 12-11, has its own job to do. Some bits cause the ALU to perform a job. Other bits specify what addressing mode should be used. Still another bit specifies the 8502 register to be used. Table 12-1 is a list of the 56 instructions. It gives a short description of each instruction and the three letter mnemonics that indicate the job. The three letter abbreviations of each instruction are the ones used when an assembly language program is written. Each three letter abbreviation is assembly language code for its binary bit set.

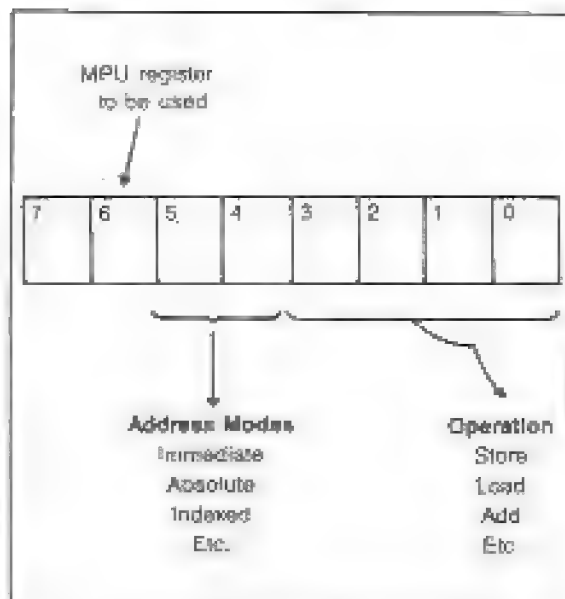


Fig. 12-11. The 8502 uses eight-bit instructions. In a typical instruction, different bits are coded to result in specific jobs to take place. For example, bit 6 could choose the MPU register to be used, bits 5 and 4 might determine the addressing mode, and bits 3, 2, 1 and 0 could specify the operation.

Fetch and Execute

When a program is run, the instruction cycle, in general, proceeds in the following way. It conducts three cycles: One is the *fetch*. Two is the *decoding*. Three is the *execution*. The entire operation is loosely called *fetch and execute*. Figure 12-12 illustrates one operation.

As the fetch begins, the program counter outputs the 16 address bits onto the address bus. The high bits select the chip that is addressed and the lower bits address the byte location on the chip. At the same time, the R/*W line is activated—a high for a read and a low for a write. The R/*W line sets up the data bus buffer in the 8502 and the buffers in the memory chips that are selected. Let's use a ROM read signal in this example. We'll fetch an instruction.

Upon being addressed the ROM chip decodes the address it received in its internal decoder stage. The decoder then sends a pulse to the location that

Table 12-1. The 8502 Instruction Set.

8502 Instruction Set			
Instruction	Mnemonic	Instruction	Mnemonic
Add with Carry	ADC	Jump to Subroutine	JSR
Logical AND	AND	Load Accumulator	LOA
Arithmetic Shift Left	ASL	Load X	LDX
Branch if Carry Clear	BCC	Load Y	LDY
Branch if Carry Set	BCS	Logical Shift Right	LSR
Branch if Result = 0	BEQ	No Operation	NOF
Test Bit	BIT	Logical OR	ORA
Branch if Minus	BMI	Push A	PHA
Branch if Not Equal to 0	BNE	Push P Status	PHP
Branch if Plus	BPL	Pull A	PLA
Break	BRK	Pull P Status	PLP
Branch if Overflow Clear	BVC	Rotate Left	ROL
Branch if Overflow Set	BVS	Rotate Right	ROR
Clear Carry	CLC	Return from Interrupt	RTI
Clear Decimal Flag	CLD	Return from Subroutine	RTS
Clear Interrupt Disable	CLI	Subtract with Carry	SBC
Clear Overflow	CLV	Set Carry	SFC
Compare to Accumulator	CMP	Set Decimal	SED
Compare to X	CPX	Set Interrupt Disable	SEI
Compare to Y	CPY	Store Accumulator	STA
Decrement Memory	DEC	Store X	STX
Decrement X	DEX	Store Y	STY
Decrement Y	DEY	Transfer A to X	TAX
Exclusive OR	EOR	Transfer A to Y	TAY
Increment Memory	INC	Transfer SP to X	TSX
Increment X	INX	Transfer X to A	TXA
Increment Y	INY	Transfer X to SP	TXS
Jump	JMP	Transfer Y to A	TYA

the address bits specify. That location is enabled while all the other locations on the chip remain turned off.

The accessing takes a few hundred nanoseconds, depending upon the access speed of the chip. The eight bits in the location are then let out on the data bus. The eight bits flash over the data lines to the 8502. They arrive at the instruction register, the IR. The IR is eight bits wide and latches the eight-bit instruction. Once the eight bits are latched in the IR, the instruction fetch is complete. A byte of data could have been fetched in the same way. The only difference is that the data would have been sent through the data bus buffer instead of being latched in the IR. The 8502 would have known what to do since the instruction would have preceded the data and the instructions would have arranged the data reception.

The decoding cycle begins as the IR releases the instruction and feeds the instruction bits into the instruction decoder. The decoder is a microscopic PLA circuit. It takes the bits and produces a set of output bits for use in the other 8502 registers, shown in Fig. 12-13.

The decoder could send one bit to the Y index register, one bit to the X index register, one bit to the stack pointer, one bit to the ALU, one bit to the accumulator and one bit to the A7-A0 low section of the program counter. It also has one bit for the input data latch. These single bits are able to turn these registers on or off according to what instruction is being processed.

The execute takes place as the decodes are driven by the clock. Each clock cycle makes the decoder perform an instruction step. The various instructions of the 8502 need different numbers of

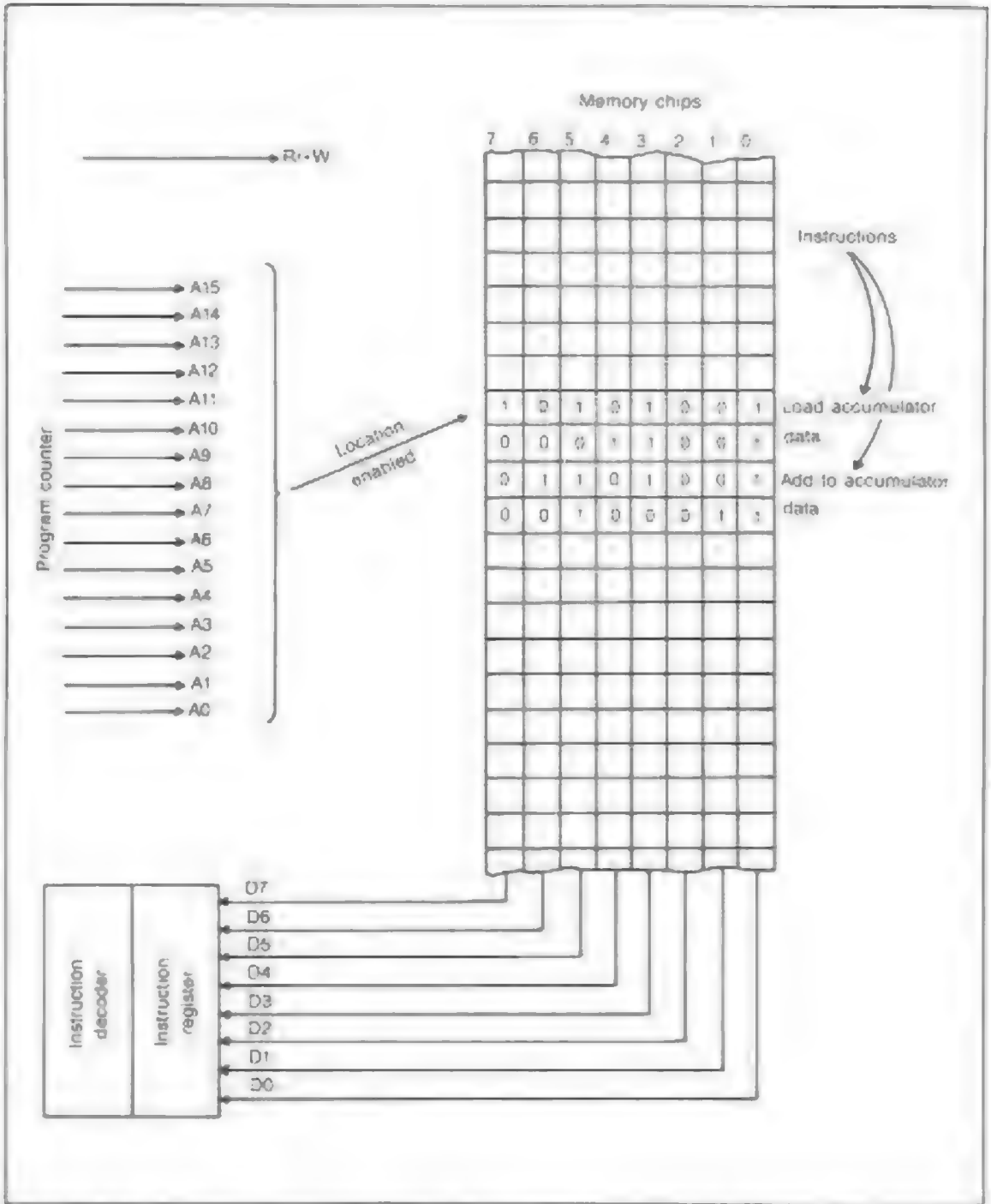


Fig. 12-12 The "fetch and execute" performance of the MPU includes addressing, data movement, decoding, and then the actual circuit execution of the instruction.

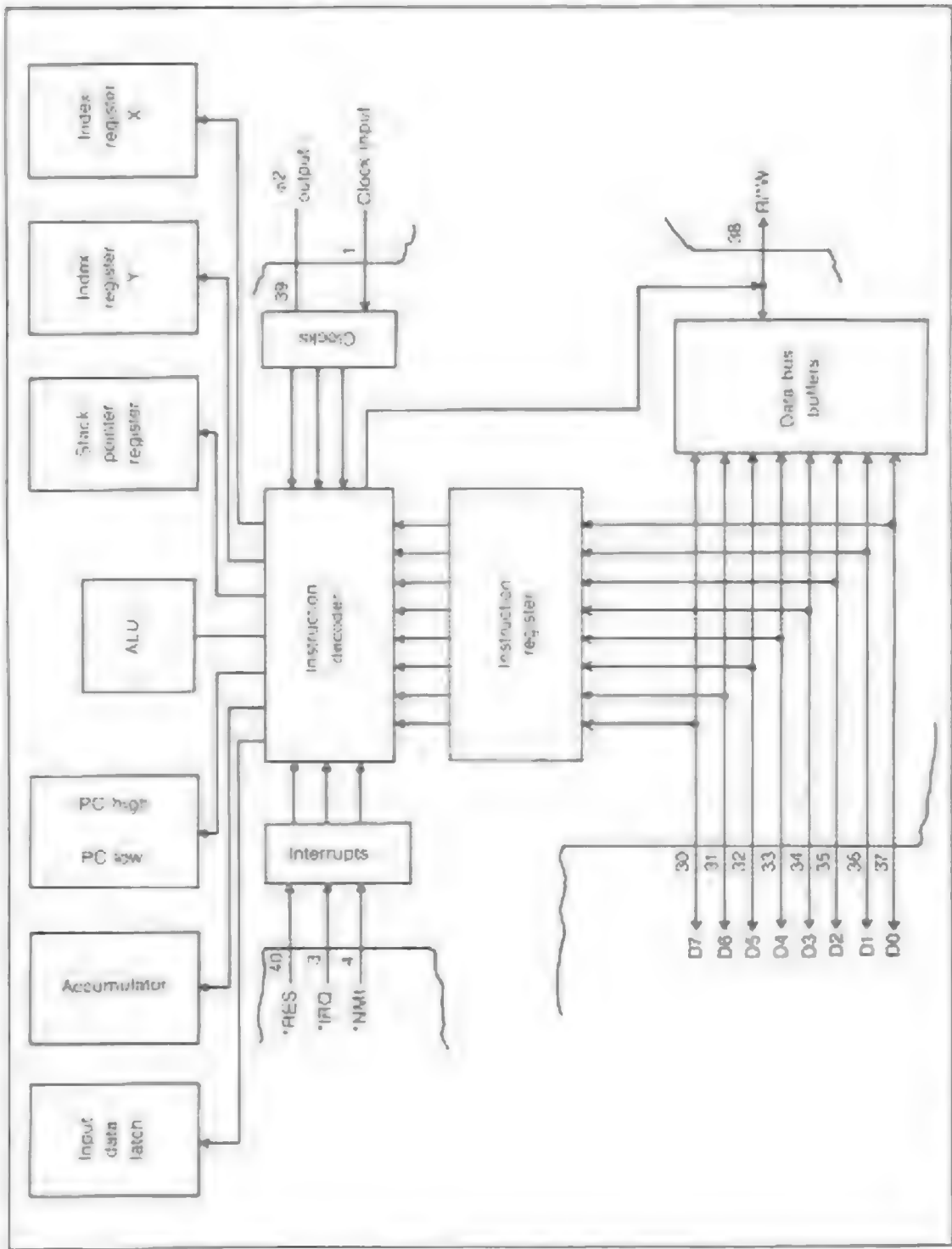


Fig. 12-13 When bytes enter the data bus, they are either instructions or data. Instructions go to the IR and data goes to the data bus buffers.

cycles to complete the instruction. Some instructions like "decrement the Y register" can be executed in two cycles. Other instructions like "arithmetic shift left" might need six or seven cycles to complete. When the 8502 is running at a 1 MHz pace a complete cycle takes 1000 nanoseconds, or 1 microsecond. When you think of timing it is usually in terms of one cycle.

Once the instruction is decoded, the various registers that are taking part in the instruction execution are turned on and the instruction is able to take place. During a read, the data follows the instruction bits on the bus. The data could be bits that need to be processed or the data could be an address that needs accessing. Whatever the data might be, it is not an instruction and goes to the data bus buffer and not to the IR.

This byte, which is after the instruction, enters the 8502's internal data bus through the data bus buffers. The data then enters the enabled registers that connect to the internal data bus and is processed. The next instruction with its data trailer is ready to be fetched and executed.

The 16-bit program counter registers have a built-in automatic incrementor. As soon as the PC outputs an address, the binary value in the PC is incremented by 1. That way as soon as the 8502 finishes one fetch and execute cycle, the next address is sent out over the address bus. The PC will always simply increment itself unless an addressing instruction like Jump, Branch, Call or Return is encountered. Then the PC will hold off on incrementing and follow the addressing instructions of the byte.

Once the special addressing instruction, such as Jump or Branch is dispatched, the PC returns to the automatic incrementing it is built to process. The PC can start at the beginning of a long program and mindlessly increment continually unless it is stopped by one of the special addressing instructions. There are only a few addressing instructions that the 8502 will respond to. There are ten Branch instructions, two Jumps and two Returns

INDEX REGISTERS

Besides the accumulator, the X and Y index registers get most of the action. A machine language

programmer uses them continually. Indexing is a valuable technique. One use of index registers is their ability to look up data in tables that are installed in the memory map. For instance, a multiplication table can be placed in a program. The address of the first number in the table can be placed in the index register. Then when a lookup is needed, an offset number is added to the table start address in the index register. The offset plus the table start number forms an address. This location contains the desired lookup. The table location is addressed by the index register and the lookup is retrieved.

The index registers can address a location. This facility and some other programming tricks make the index register valuable. The 8502 has two eight-bit index registers, called X and Y. They are used almost as often as the accumulator. In fact, the index registers, besides their tricky addressing capabilities, can be used as scratch pads, to store data temporarily, just like the accumulator.

The accumulator and the index registers are the link to RAM locations. You cannot store data directly into RAM. In order to store data in RAM you must first load the data into the accumulator or the index registers. Then they will store the data for you.

The above discussion reveals what is happening in the 8502 and the things the index registers can do. If you want to check an index register you can do it with a little BASIC routine. For example, when you write a program with FOR . . . NEXT statement around a POKE, you are creating a loop.

```
100 FOR Z=1024 TO 1054
200 POKE Z,0
300 NEXT
```

The BASIC ROM in turn sends out machine language bits that do the following to accomplish the FOR . . . NEXT loop. First, it loads the accumulator with the data that is to be POKEd. Then it transfers the contents of the accumulator to the X register. Then it stores the contents of the accumulator in address 1024 + X.

The first time through the loop, the 0 is stored in 1024. The next machine language instruction is: increment the X register. Once incremented, the X register becomes 1. After that the X register is compared with the decimal number 31 (1054 - 1024 + 1). If the contents of the X register, at that time, does not equal 31, then the loop continues. Because X contains 1, the loop continues. When it finally does equal 31 the registers 1024-1054 are loaded and the routine stops.

The above shows what happens between the accumulator, the index register, and RAM when you write a FOR . . . NEXT routine in BASIC. To sum it up, if you want to test the X index register in the C128 write a FOR . . . NEXT loop and see if it executes. If it does, then you have exercised the accumulator, the X index register, and a section of RAM that you might have POKED data into. They are all apparently okay if the loop works.

During repair work, the index registers are not usually a problem. They are part of the 8502 block

diagram though, and a servicer should be familiar with their operation.

FLAG REGISTER

The flag register is shown in Fig. 12-14. It is also known as the Status Register or Condition Code Register. It is a special eight-bit register. It is actually eight single bit registers bunched together. Each of its bits does its own job. This register in the 8502 has eight bits, but only seven of the eight are in use. Bit position five is unused. All of the rest have individual jobs.

The bits are called flags. The flag can be in one of two states, 1 or 0, on or off. When a flag is set to a 1 it is in use. A 0 bit means the flag is lying still.

The flags are vital during a program run. As the program is progressing from line to line and instruction after instruction, the flags are continually turning off and on according to the instruction. Most of the 8502 instructions cause one or more flags to

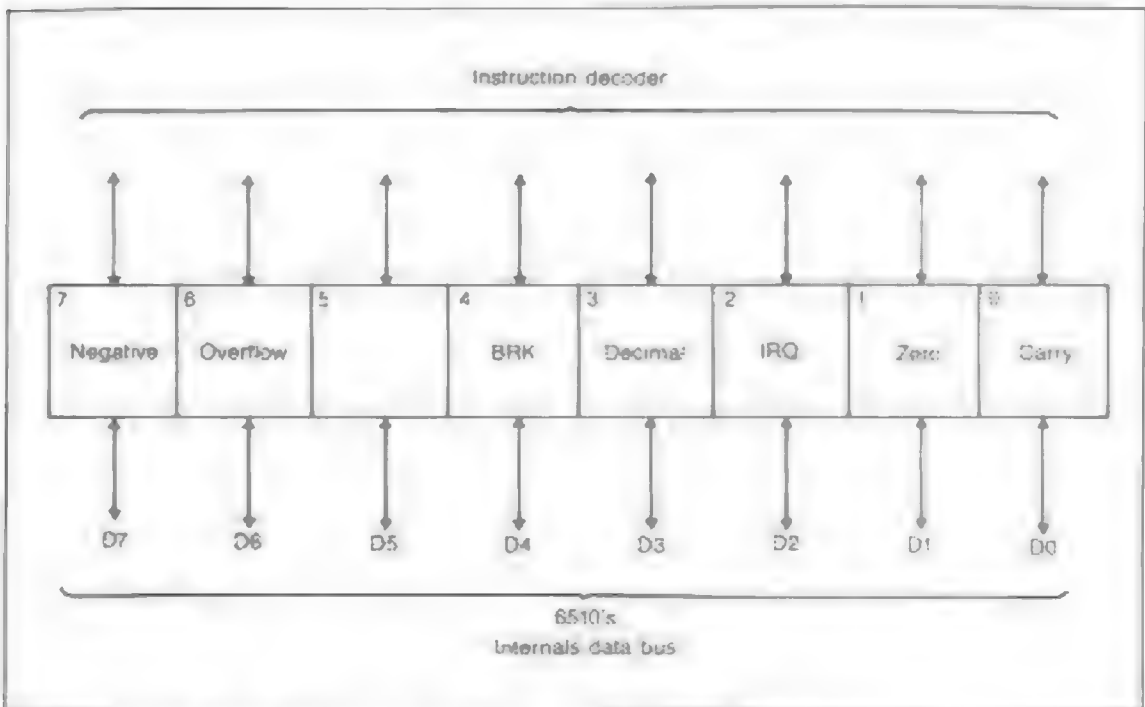


Fig. 12-14. The flag register, also known as the status register, is connected between the data bus and the instruction decoder. Each flag is a single-bit register. The flags monitor every decoding operation and are set or cleared accordingly.

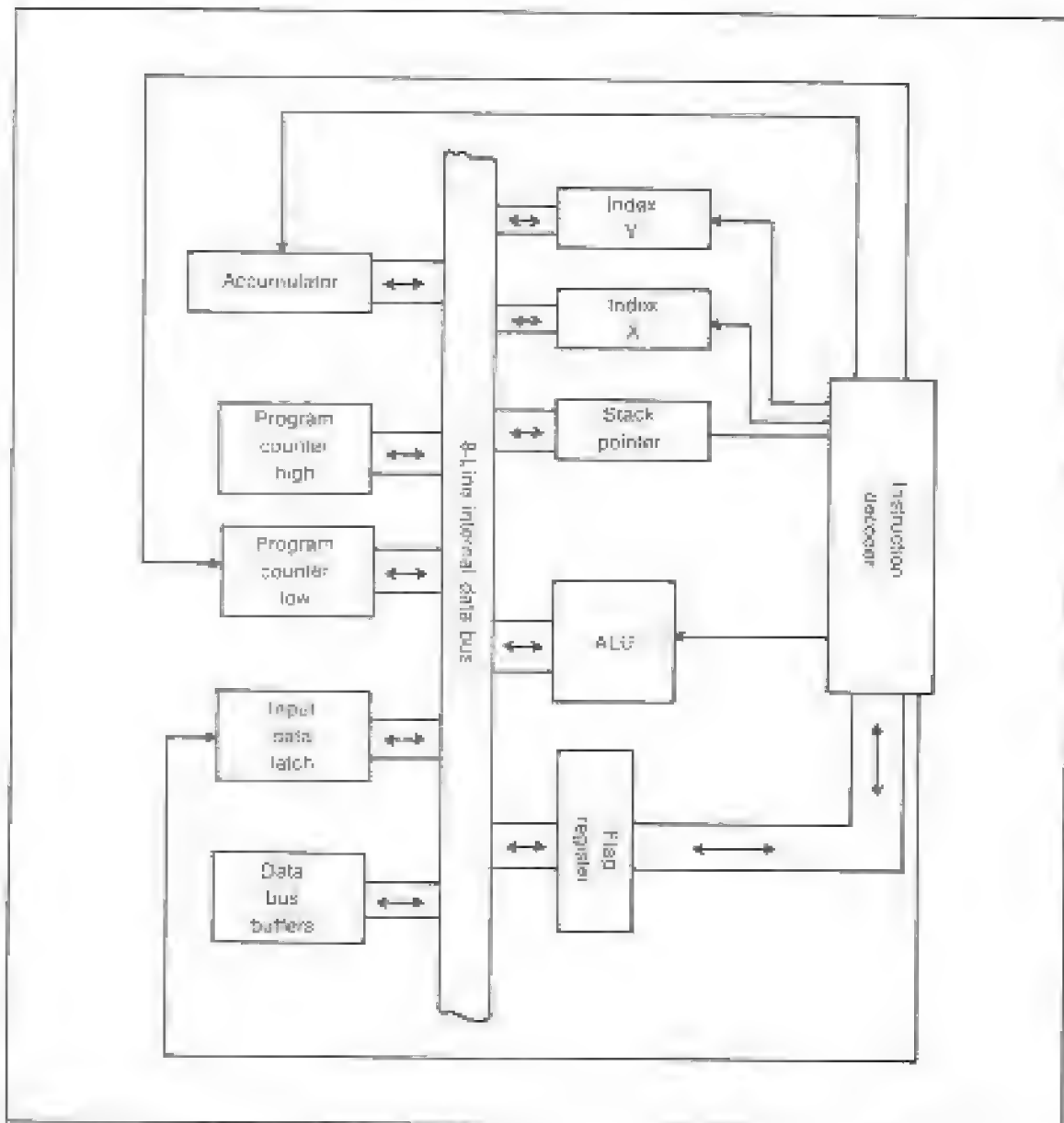


Fig. 12-15. The flags do most of their business with the ALU.

make a change. If the flags were attached to lights, the lights would be flashing continually as a program is run.

The machine language programmer must be completely aware of every flag that gets set or cleared. Each flag movement performs some sort

of duty and has important meaning. The flag register is connected to the instruction decoder through one eight-bit bus and to the 8502's internal data bus through another eight lines, as seen in Fig. 12-15. The flags conduct most of their business with the ALU through the internal data bus. The main job that

the flags do is to set when the accumulator attains a particular state. Let's examine some of the states that the accumulator can get itself into.

Programmer's View of the Flags

The flag register has its eight bits arranged as in Fig. 12-16. Note that this block diagram shows all of the important 8502 registers except for the instruction register, the data bus buffers and the ALU. This is because, as a machine language program is written, only the accumulator, index, stack pointer, program counter and flag registers are considered. The IR, buffers, and the ALU do not enter into the programming. However, for hardware considerations such as troubleshooting, all the registers must

be considered, because any of them can fail and bring down the machine.

This programmer's block diagram shows the size of each register in bits. Note that the accumulator and index registers are eight bits wide. The program counter, between its PC high and PC low, forms 16 bits. The stack pointer is eight bits plus the ninth bit: set at 1 to place the stack at addresses 1 0000 0000 (256) through 1 1111 1111 (511). The flag register is shown as eight individual bits with bit 5 blank. In the other bits are the letters, N, V, B, D, I, Z and C.

Bit 7 of the flag register contains the N flag. N stands for negative. The negative refers to the numeric state of the contents of the accumulator reg-

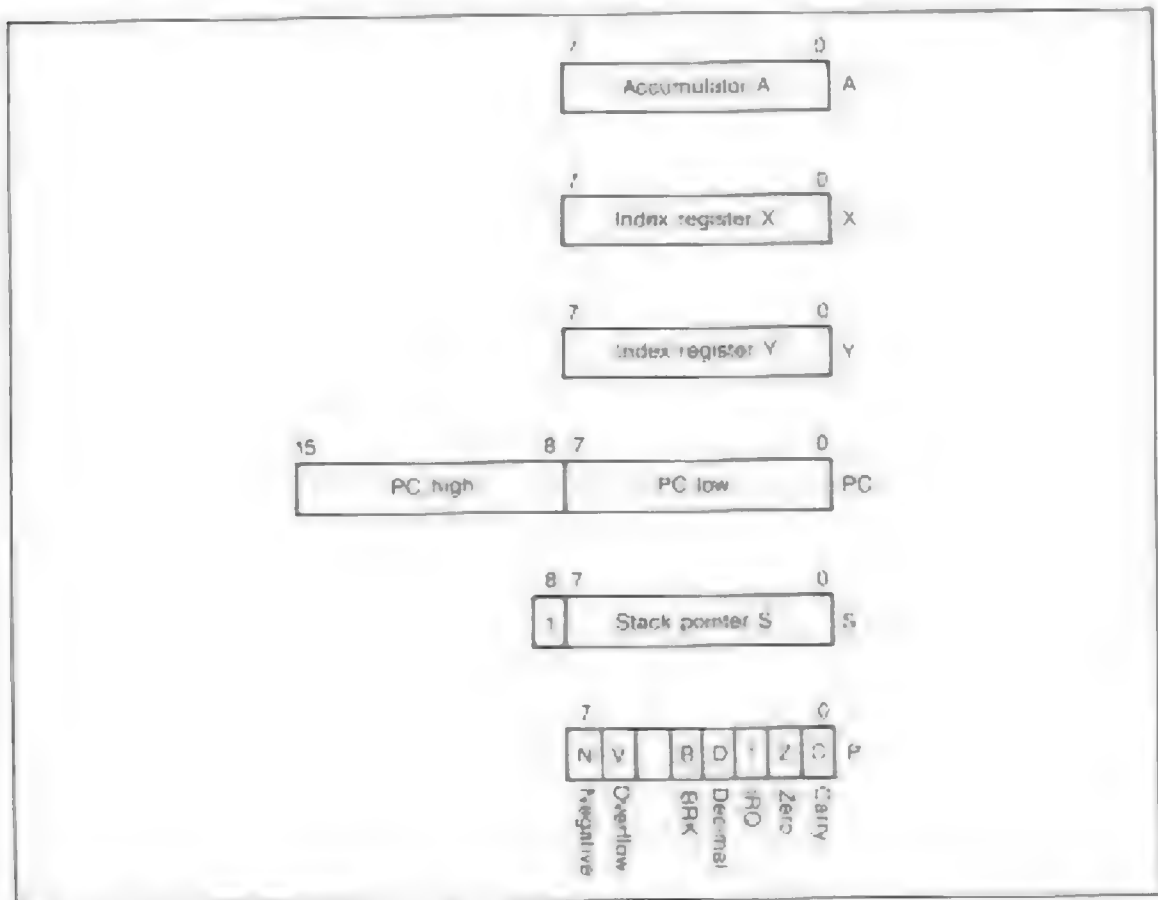


Fig. 12-16 This is a block diagram used by the machine language programmer. It shows the programmable registers and their bit sizes.

ister. The accumulator is also an eight-bit register. During calculations that use negative numbers, the accumulator must hoist a flag to show that its contents are a negative number.

When the negative number is in the accumulator, the accumulator's bit 7 becomes a 1. When the accumulator has a positive number, its bit 7 becomes a 0. Therefore, whenever the accumulator bit becomes a 1, the N flag is hoisted—that is, it becomes a 1. Otherwise the flag stays at a standby 0 state.

This arrangement works well for the programmer but can get complicated. Nothing's perfect. The complication is that the flag gets set whenever the accumulator bit 7 becomes a 1, no matter what. This is fine for calculations but can prove to be an unwanted flag raising in other programs, for instance word processing. However, this is a problem for the programmer. All a tech needs to know is that the flag will be set whenever bit 7 of the accumulator goes to a high.

One use of the N flag is to cause a change of address when the accumulator contents become a negative number. This ability is a handy programming tool.

Bit 6 is the *Overflow* flag. It is used in two's complement calculations (a programming device). Repair jobs have very little need for two's complement figuring. For repair purposes just be aware that the Overflow flag gets set when there is an arithmetical carry from bit 6 to 7 of the accumulator.

When this overflow occurs, a special correction routine must be run off by the 8502 to correct the overflow. The reason it is important to note this bit 6 to 7 overflow is that the overflow can change bit 7's negative or positive sign and harm the program run. The overflow must not remain uncorrected.

Bit 5 of the flag register is unused in the present 8502. That does not mean it will always remain unused. Future models of the 8502 could include a bit 5 that does some sort of job.

Bit 4 is the *Break* flag. There is one instruction called *Break*. The instruction, as it arrives at the IR and becomes operational, will set bit 4. This break effect is useful during program writing and running.

Bit 3 is an arithmetic helper. When bit 3 is not set, the 8502 runs its mathematics normally in bi-

nary. During some decimal operations, however, the binary mode of counting from 0 to 15 is clumsy. It is easier in these cases to be able to count from 0 to 9. This computer ability to count in decimal instead of binary is called Binary Coded Decimal, BCD. The 8502, when bit 3 is set, has the valuable ability to cease counting in binary and start counting in decimal. When the flag is cleared, the 8502 goes back to counting in binary again.

Bit 2 is the IRQ flag. IRQ stands for *interrupt request*. It is referred to as the *interrupt mask bit*. When this bit is set, the interrupt is masked and no other interrupt can interrupt. The bit can be set with some instructions by the programmer, or the 6802 itself will set it during a RESET operation. An interrupt will also set the bit.

Bit 1 is the zero flag. The zero flag like the N flag can be set under many conditions. It becomes set whenever the accumulator goes to a state of all zeroes. The Z bit is built to get set whenever the accumulator 0 state occurs—intentional or not. That is the programmer's problem.

Bit 0 is the Carry flag. It is that ninth bit that was mentioned earlier in the accumulator shift and rotate discussion. It will receive the bit that gets carried out of bit 7 of the accumulator during arithmetic. It will also catch the bit that falls out of the accumulator during shift and rotate operations. For more details on flags, refer to the many books on 6502 machine language. For repair purposes, the above knowledge is sufficient to run tests on the hardware.

More About Interrupts

In the 8502 schematic diagram of Fig. 12-17, are three interrupts that can enter the instruction decoder and cause the 8502 to stop what it is doing and service the interrupts. They are called RESET into pin 40, IRQ into pin 3 and NMI into pin 4.

The RESET circuits, Fig. 12-18, are used to get the C128 started. When the machine is turned on, +RESET is applied to pin 40 of the 8502. +RESET begins life when the RESET button is pressed and closes. This puts +5 volts into pin 8 of U27, the 556 timer chip. The timer in response outputs a low from pin 9. The low passes through a NOT buffer stage of U63. This conditioned low

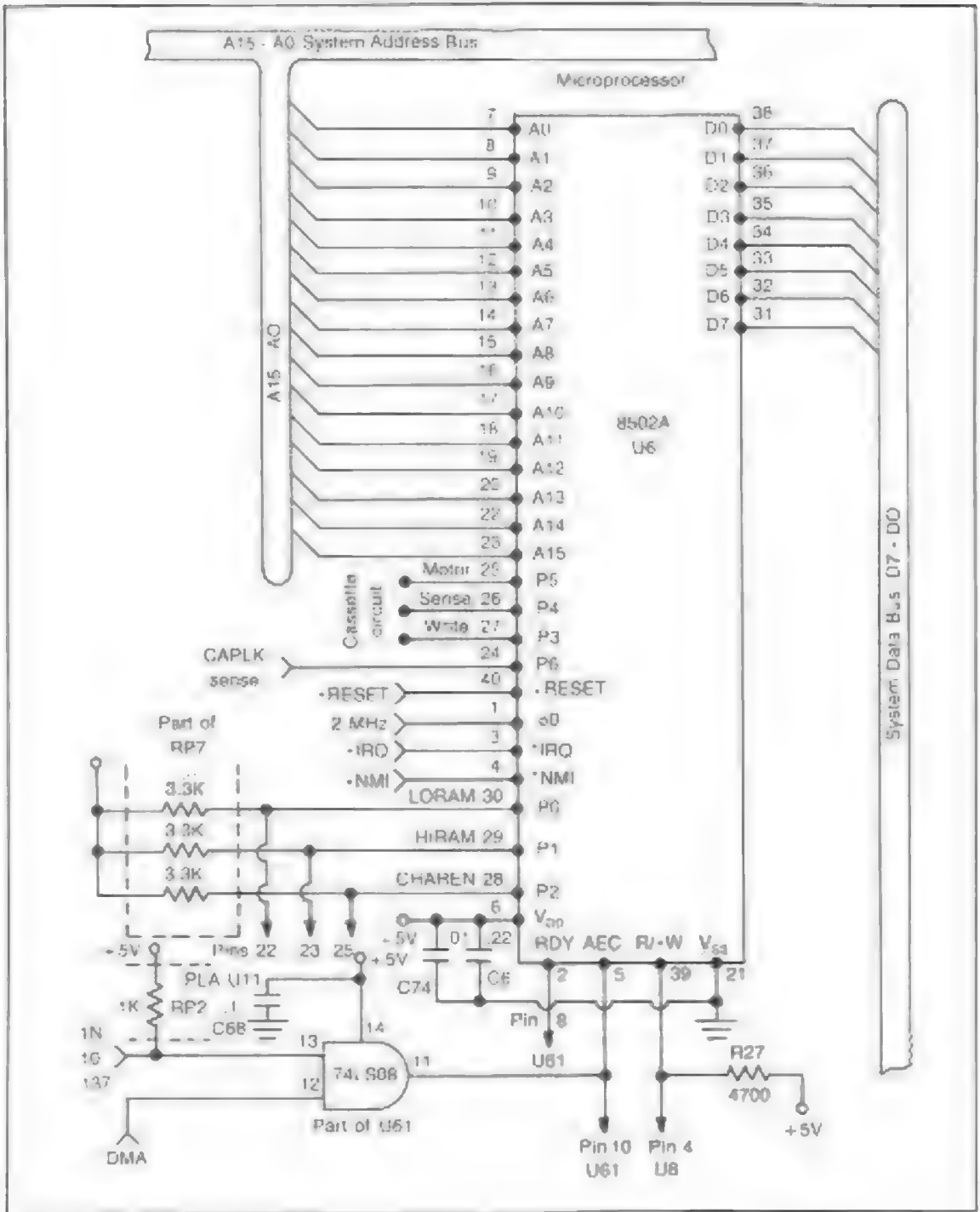


Fig. 12-17 Schematic diagram of the 8502 processor

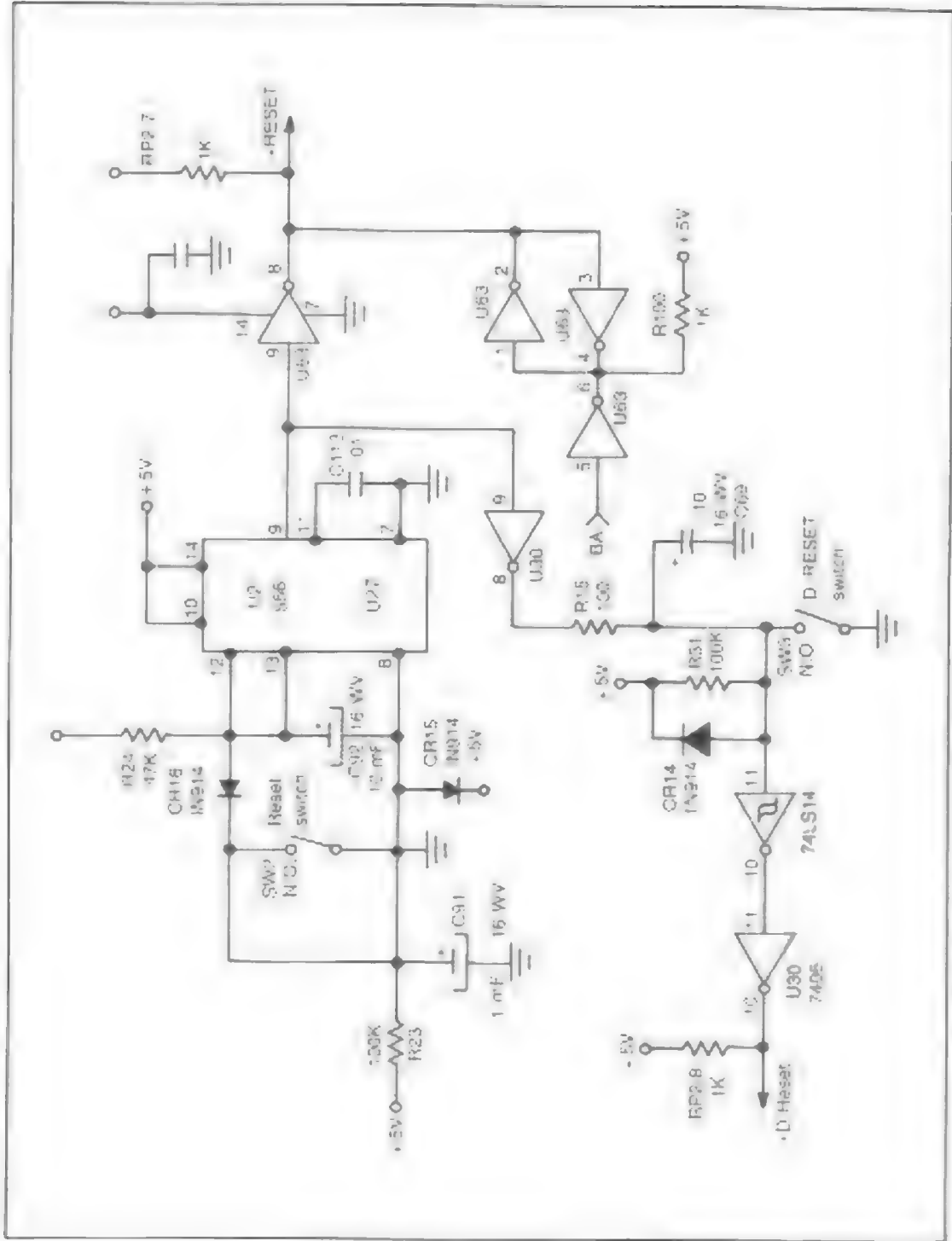


Fig. 12-18 The reset circuit starts the C128 from a cold start. The circuit can also reset the machine when the Reset Switch is pressed

is $\overline{\text{RESET}}$. $\overline{\text{RESET}}$ goes to a number of chips including pin 40 of the 8502.

While the RESET button is held down, the $\overline{\text{RESET}}$ low stops all activity coming from or going to the 8502. Then, when the RESET button is freed, the low changes back to a high and the 8502 starts working anew. The RESET routine begins. The first thing that happens is: the 8502 registers are initialized and the R/W line is made high so that the 8502 can read from memory. Then the mask interrupt flag is set so no other interruptions can interfere with the RESET operation.

The 8502 then reads two eight-bit locations from the Kernel ROM. These two bytes are called a *vector*. The vector is an address. The 8502 loads this address into its program counter and the address goes out over the address bus. The address is the starting location of the operating system's initialization program.

The program is then run off and the screen soon shows READY and the cursor is blinking. That is what the RESET input into pin 40 makes happen.

The second interrupt line into pin 3 is $\overline{\text{IRQ}}$, interruption request. It is normally held high and becomes active when forced low. When the low arrives, the 8502 goes into the interrupt routine. This interrupt typically happens when a peripheral device calls the 8502 through an I/O port. The peripheral usually wants to send or receive data. The $\overline{\text{IRQ}}$ line comes from pin 21 of CIA1 and pin 4 of the expansion bus.

When the $\overline{\text{IRQ}}$ signal arrives at the 8502, the first thing the 8502 does is to complete the instruction it is working on before it acknowledges the $\overline{\text{IRQ}}$ signal. Once it does complete the current instruction, it checks flag bit 2, the IRQ bit. As long as the mask is not set, the 8502 will then service the interrupt. Should the bit be set (the interrupt is masked), the 8502 will ignore the request.

Once in action on the interrupt, the 8502 stores some of its registers in the stack. The program counter's current address goes into the stack. The states of all the flags are stored in the stack. The 8502 then goes to the flag register and sets the IRQ flag so no other interrupt can interrupt the service program that is about to take place.

The 8502 is built so that, at a time like this, it addresses the last two addresses on the 64K memory map: 65534 and 65535. These addresses are part of the Kernel ROM. They are also known as the $\overline{\text{IRQ}}$ vector. In these vector addresses is another address in the Kernel. This address is the start location of the IRQ service routine. The 8502 reads this address and loads it into the program counter. The program counter in turn addresses the service routine and the 8502 proceeds to run off the routine.

As we humans follow these interrupt procedures, they seem complicated and difficult to follow. The C128, though, performs this game of using address after address to finally find the service routine as second nature and has no difficulty. It whips through it in millionths of a second.

Once the interrupt has been serviced and the peripheral that caused the interrupt has been satisfied, the 8502 goes back to what it had been doing before the IRQ had arrived. The 8502 goes back to the stack and places the program counter and flag values back into the PC and flag register. A special interrupt return instruction is executed and the 8502 resumes its program by addressing location after location.

The third interrupt is called $\overline{\text{NMI}}$ for non-maskable interrupt. It is activated by a low into pin 4 of the 8502. It works quite like the $\overline{\text{IRQ}}$ interrupt. Both interrupt pins, upon receiving a low, start executing. Both preserve the contents of the PC and the flag register in the stack. Both automatically address two locations near the top of memory.

The difference is in their relationship with bit 2 in the CC, the flag register. Bit 2 will mask off an $\overline{\text{IRQ}}$ when it arrives if bit 2 is set. Bit 2 will only permit an IRQ if it is not set. $\overline{\text{NMI}}$ is different. The $\overline{\text{NMI}}$ will always go into action no matter what bit 2's state happens to be.

In many computers, the $\overline{\text{NMI}}$ is used in case there is an emergency interrupt like a power failure. In the C128 however, it is used for a different purpose. It is used in the RESTORE key circuit.

When the RESTORE key is pressed, as Fig. 12-19 shows, it sends signal RSTR to pin 6 of U27—the 556 chip that is also used for the RESET activity. Out of pin 5, a signal emerges and enters

a section of U29, a buffer NOT gate. The \bullet NMI signal then proceeds to pin 4 of the 8502.

When \bullet NMI arrives as a signal from the RESTORE key, the processor goes through the same sort of sequence as if an IRQ had arrived, except that it loads the program counter with a vector address from two bytes near the top of the memory map in the Kernel ROM.

The term *vector* is confusing. Here, it describes two byte-sized locations whose combined contents form one 16-bit address. For instance, Kernel ROM locations 65535 and 65534 together contain the vectoring address of the start location of the \bullet IRQ ser-

vice routine. Location 65533 and 65532 contain the start address for the \bullet RES routine. 65531 and 65530 will be addressed when \bullet NMI in the processor is turned on.

This interrupt operation is not unlike a game where you are given an address. At that address you will find another address that will point you to the prize, which in this case is the little program in ROM that services the interrupt.

THE 8502 SIGNALS

The 8502 has 40 pins, as seen in Fig. 12-17. There are 16 pins coming from the program counter,

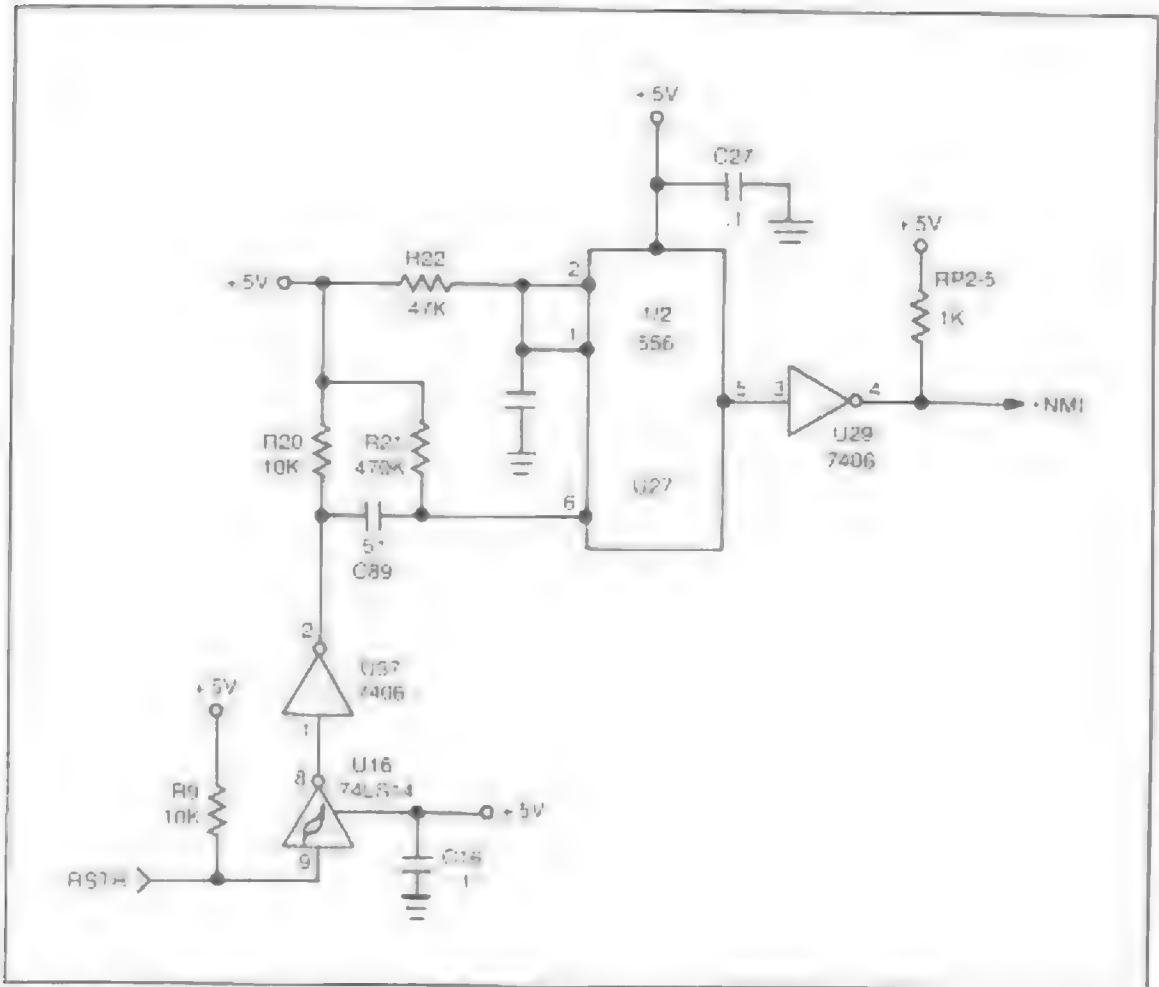


Fig. 12-19. When the RESTORE key is pressed, the \bullet NMI circuit sends a signal to the 8502.

A15-A0, that output the address bits. There are eight pins connected to the instruction register and data bus buffers that both input and output data bits. Then there are the three forms of interrupts, IRQ, RES and NMI that receive interrupt inputs. There are two pins for +5 volts and ground. That takes care of 29 of the 40 pins. What about the remaining 11 pins? What sort of computer signals are they acting as ports for?

There are seven pins, 24-30, called P6-P0. P6 is an input from the CAPS LOCK key on the keyboard. When you hit the key it will lock and send a signal to P6. In upper/lowercase mode, the processor will then only permit capital letters to be placed on the screen. P5, P4 and P3 are three connections to the cassette circuits. P5 operates the cassette motor, P4 the cassette sensor and P3 the cassette-write impulses. P2, P1 and P0 are connected to the PLA. They connect the signals LORAM, HIRAM and CHAREN to the PLA.

At pin 5 is the Address Enable Control, AEC. This is an important input from VIC. The 8502 shares the use of the address bus with VIC. When VIC is permitting a high at pin 5, the 8502 is in control of the address bus. Should VIC send a low, though, the 8502 is effectively turned off and VIC uses the address bus.

At pin 2 is RDY, "ready." As long as RDY is high, the processor performs normally. When RDY goes low, the processor will finish the operation it is conducting and then turn off and let another circuit take over the bus lines.

Pin 39 is the all-important read/write line called R/W. As the name implies, when the line goes high the read operation takes place. When the line is forced low the write operation goes on.

Finally, Phase ϕ is input from the clock circuits at pin 1. This is the signal that drives the 8502. Both the 1 MHz and 2 MHz clocks are input at this pin.

TESTING

Fortunately, the 8502 is a very reliable processor and is not the culprit when the computer fails. However, like any electronic gear, it is subject to

breaking down. The following tests can quickly check it out.

One quick check can be made with the C128's Monitor operating system. In the C128 mode, all you have to do is type the direct command MONITOR or press F8 (if this key is so programmed) on the keyboard and you should see a display like the following.

```
MONITOR
  PC  SR  AC  XR  YR  SP
: FB00 00  00  00  00  F9
```

Different C128's might not be exactly like this one, which is off of my C128 display. If your display comes up in this way, the odds are very good that your 8502 is working fine. This display shows the contents of the 8502 registers as you entered the monitor. The contents are described in hex. The program counter, PC, is about to address the first machine language instruction. The status register, SR, has all eight flags with 0's. The accumulator, AC, is initialized with all 0's. The X and Y index registers, XR and YR, are both initialized with 0's. The stack pointer, SP, shows the first address on the stack in hex. The 8502 is ready for action. The action is entering machine language programs.

The Monitor is nothing more than an operating system, just as BASIC is. The Monitor though lets you program with the 8502 instruction set and deal directly with the 8502 programming registers. This gives you the opportunity to keep a close watch on the registers for troubleshooting purposes. If the registers start showing hex contents that are not predictable, that could be an indication of a bad acting 8502. The Monitor also works along with RAM, ROM, VIC and other chips. It could provide answers to repair puzzles.

Of course, a look at the Monitor can only happen when the circuits are working enough to respond to the MONITOR command. If the C128 is not computing enough to show the Monitor responding to the command, then you must grab the logic probe

TEST POINT CHART
U6
8502 MICROPROCESSOR

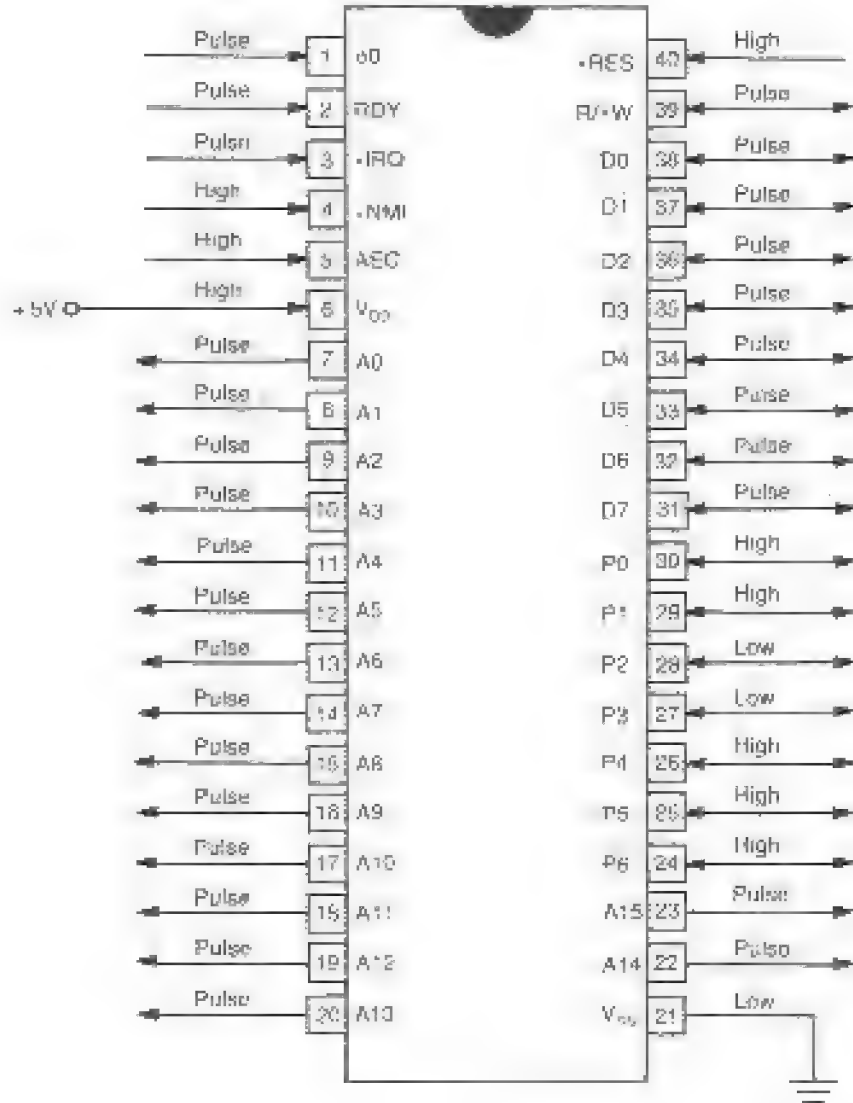


Fig. 12-20 These are the logic states that should be on the 8502 in C128 mode and timing.

or vom and start testing. Figure 12-20 is the Test Point Chart for the 8502. It shows what should be present at each pin.

The chart readings were made on my C128 with the machine energized, the READY sign on, and the cursor blinking. This is a good time for pin-by-pin readings. All of the Test Point Charts were made under the same operating conditions.

You can assume all is well with a chip under test as long as the pins read the prescribed voltages or states. If a discrepancy is found, then that is a clue and bears further investigation.

In general, if you find an incorrect reading, then note if the pin is input or output. If the wrong reading is on an input pin, chances are that the chip is okay and the trouble is located in circuits providing the input signal. When the wrong reading is on an output pin odds are that the trouble has happened inside the chip and the chip will probably need replacement. This is not a hard and fast rule, be-

cause, on repair jobs, some of the oddest complications can and do occur. It is a good rule to start the repair with, however, and most of the time will prove out.

The service charts are drawn around the actual pinout of the chip. If you find a problem pin, you can then go to the schematic, Fig. 12-17, which does not follow the pinout, and see where the signals and voltages are coming from.

You'll find that the address and data pins all will show PULSE on the logic probe. Pin 1, the clock signal input, also has PULSE.

The three interrupt inputs should all read HIGH till they are enabled, and receive a low. On normal standby, however, they are high. RDY should show a HIGH as well as the R/*W line. The R/*W line is in a read mode when idling. V_{DD} should always be high or +5 volts and V_{SS} low or 0 volts. If any of them are not, then you have uncovered a clue that could lead to the cause of the trouble.

13. Z80 Coprocessor

The C128 is a C64 and an 8502 machine language computer, but is also a Z80 computer. The only problem with the Z80 function is that there is no operating system for the Z80 on board—it may not operate on its own. Commodore supplies a disk, with every C128 it produces, that contains a CP/M Plus™ Version 3.0. This operating system, produced by Digital Research Inc., is a popular one that runs thousands of CP/M based programs.

When you want to use the Z80, all you need to do is load the operating system from the disk to RAM. You then will have a Z80 computer with 59K of RAM available, ready to go. The CP/M may be displayed out of the 80-column Video Controller chip. If you use a 40-column monitor, you can see all 80 columns by pressing the CONTROL key and scrolling with the appropriate CURSOR key.

While the 8502 is the main processor, the Z80 is the copilot. The two of them work together. During the C128 and C64 modes, the 8502 is in charge but uses the Z80 for some purposes, such as starting the machine on power-up. When the C128 is put

into CP/M mode, the Z80 is mostly in charge but does use the 8502 for jobs like accessing the Kernel ROM routines, and some I/O work. The two processors use each other when necessary.

When power is first turned on, in any mode, the Z80 is given control. If the 8502 had been in control at the start, the Z80 could have accidentally started addressing and moving data. This would cause the program to crash from the start. With the Z80 in charge it can do whatever it wants to on the bus line. The 8502 just sits by and waits. After performing some housekeeping and initializations, the Z80 starts up the 8502 in either C128 or C64 mode—that is unless a CP/M disk is in the drive and gets loaded into RAM. When that happens the Z80 remains in charge and is ready to run or write CP/M programs.

Z80 BLOCK DIAGRAM

The Z80 has many registers. Eighteen eight-bit and four 16-bit registers are shown in Fig. 13-1. These are all accessible to the machine language

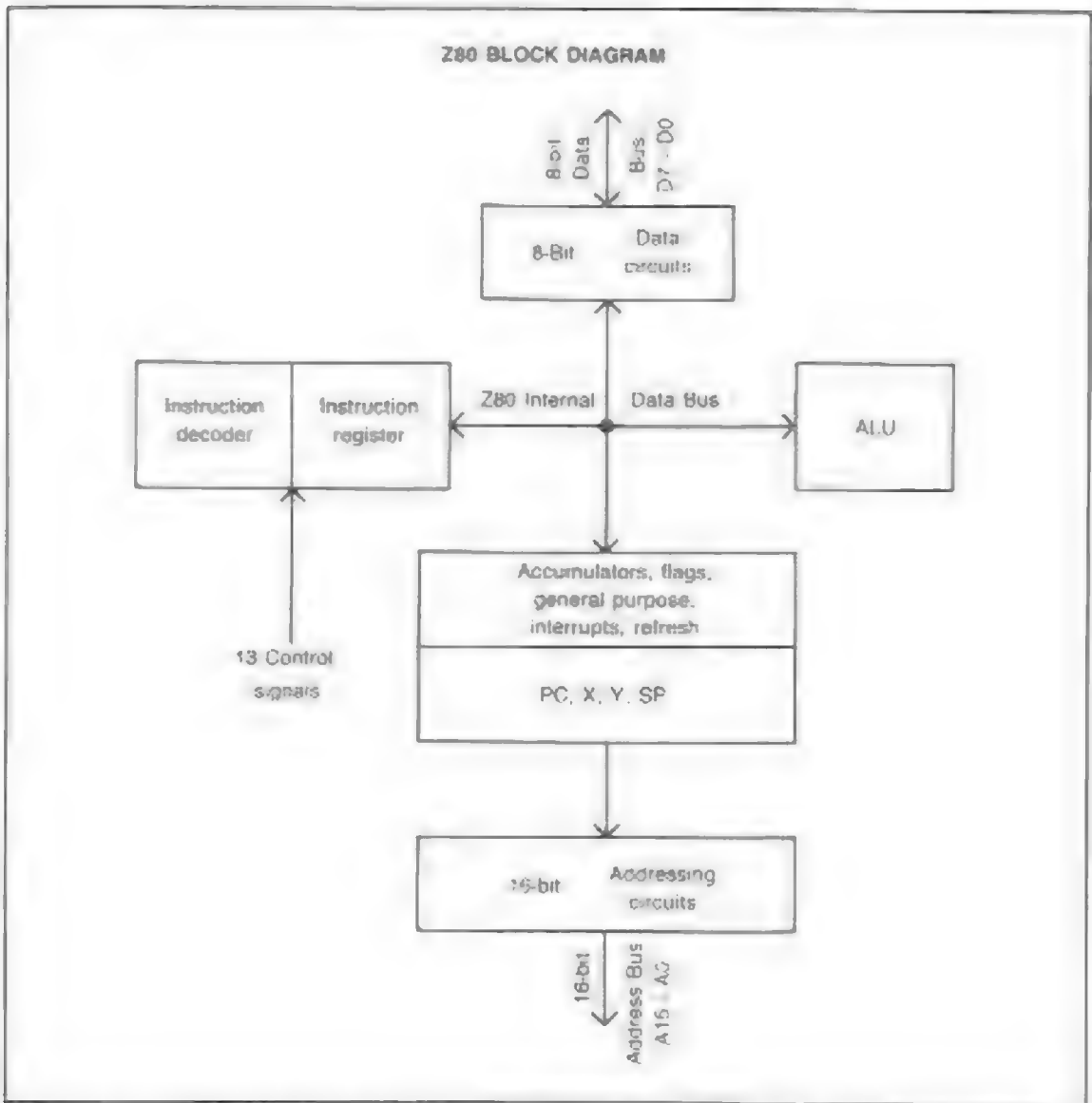


Fig. 13-2 All of the registers are in one section of the Z80. The four 16-bit registers deal in addressing and output bits over the address bus. The eight-bit registers work with data. They input and output bits over the data bus.

The X and Y index registers are two independent 16-bit registers that hold the special addresses needed for the index modes. The stack pointer is another 16-bit address that holds the start address of the stack. The stack in RAM is of the last-in first-out variety. Therefore, if there is anything in the

stack, then the stack pointer register will hold the address of the last byte that was placed in the stack.

The Z80 has two registers that the 8502 does not have. One is the eight-bit interrupt vector. This register holds the address of a memory page where an interrupt service routine is stored. During some

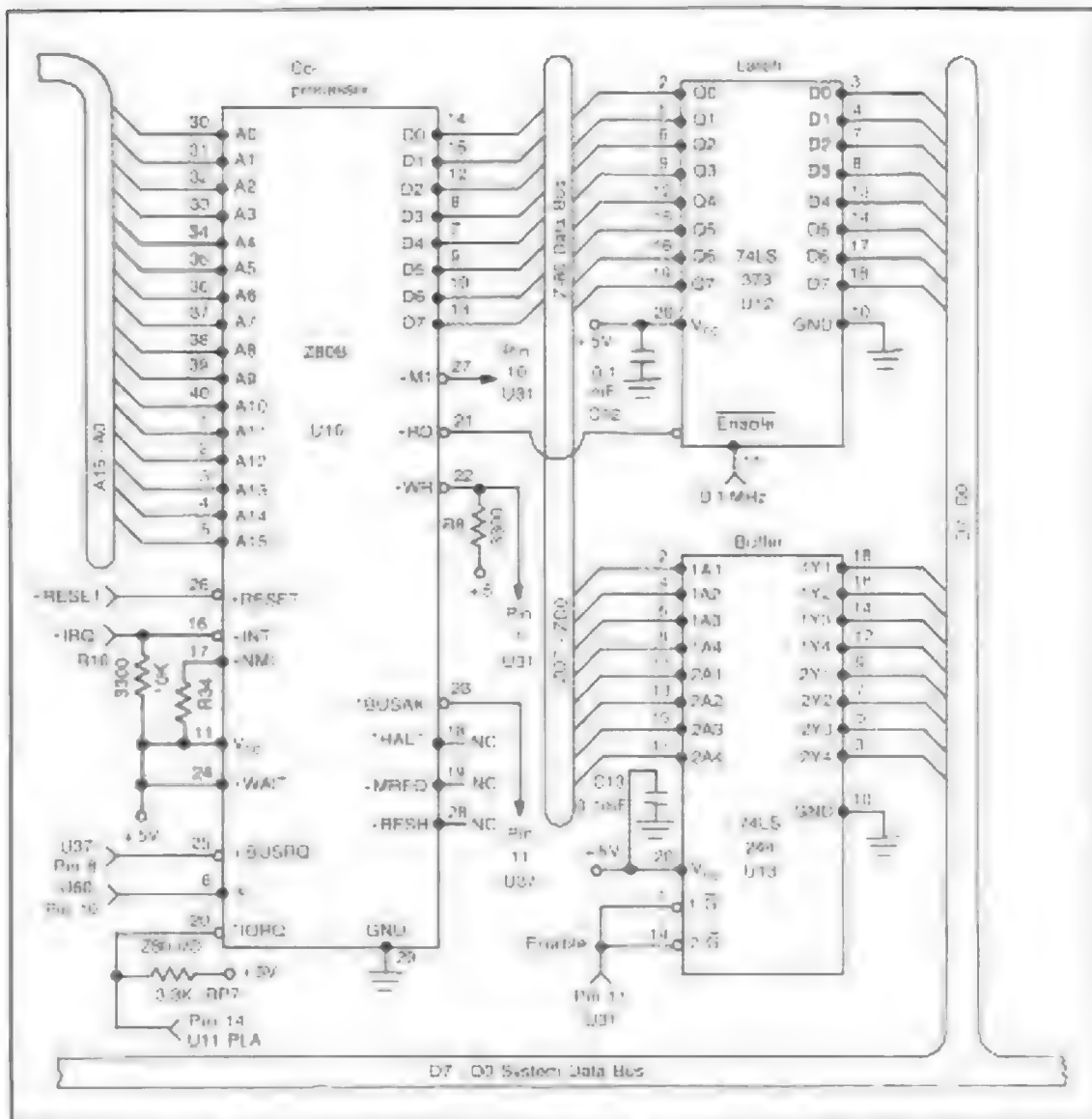


Fig 13-3. The Z80 can connect directly to the system address bus. It cannot connect directly to the system data bus. It must use the latch as an interface during a read and the buffer as an interface when performing a write.

interrupts, this vector will be placed on the address bus to address that desired page.

Then the Z80 has an eight-bit memory refresh register. This is a seven-bit address that is designed

to refresh the RAM rows after every instruction is completed. Forget about it for the U128, however, because VIC takes care of refreshing the DRAMs, and this output from pin 28 is not used.

The Non-Programmable Registers

The ALU handles the arithmetic and logic calculations that go on in the Z80. Specifically, the ALU performs the following operations: it ADDs, SUBTRACTs, ANDs, ORs, Exclusive ORs, SHIFTs LEFT and RIGHT, INCREMENTs, DECREMENTs, COMPAREs, SETs a bit, RESETs a bit and TESTs a bit. These are all machine language programming concerns and do not enter into a troubleshooting job, but the understanding that these jobs are performed by the ALU could help solve a repair puzzle on some occasions.

The other non-programmable registers are in the instruction register realm. The instruction register, first of all, is loaded from the system data bus, D7-D0, during the instruction cycle. Then the IR uses a control register to get the defined instruction performed. This includes generating all the control signals that will send the results of the instruction operation to correct register destinations.

Z80 SCHEMATIC DIAGRAM

Figure 13-3 shows the schematic diagram of the Z80 in the C128. Note that three control pins are not connected: pins 18, 19 and 28. Pin 18 is a signal from the Z80 called *HALT. This pin is not connected and will show a high. It goes low when the processor is awaiting a non-maskable interrupt which does not happen in the C128. Pin 19 is also a signal from the Z80 called *MREQ. This signal is a low when the Z80 is indicating a valid address on the address bus. *MREQ is a three-state signal and is tristating so there will be no logic probe reading present on the pin. Pin 28 is the *RFSH signal mentioned earlier. Because the DRAM refresh is conducted by VIC, the pin is held high and is inactive.

It is necessary during troubleshooting to have an understanding of the unused pins as well as the operating pins. If an unused pin's state should be incorrect then that is a good clue.

That leaves ten control pins. The first four are System Control. Pin 27, *M1, is an output that is an active low, so it will read high while the C128

is idling and being tested with the logic probe. When it goes low, it signals that the current machine-cycle fetch is in operation. Pin 20 (*IORQ) is active low and will show a high when idling. The *IORQ shows that the lower half of the address bus is valid when low. It also plays a part with M1 when an interrupt acknowledge is needed.

Pins 21 and 22 are *RD and *WR. When 21 is not in the CP/M mode it is tristating because it is a three-state output active low. When low, it is the read part of the R/W circuit. Pin 22 is the counterpart of 21 and is also a three-state output active low. Pin 22 will show a high when idling though because it is held high with a connection through a 3300 ohm resistor to +5 volts. *RD is the signal in CP/M mode that indicates when the Z80 wants to read from memory or an I/O device. *WR is the signal that indicates Z80 wants to write data to memory or an I/O device.

The next four control signals are inputs from the rest of the computer to control the Z80. Pin 24 is called *WAIT. It is an input active low. When active it tells the Z80 that the memory, or I/O device, that is addressed is not ready to transfer data. However, this circuit does not use this input and simply ties pin 24 to +5 volts to keep it high, and not have the Z80 accidentally go into a WAIT state. Pin 17, *NMI, the non-maskable interrupt pin, is also unwanted in the C128 and gets the same +5 volt connection. *NMI is active low and is thus held high and cannot interrupt the operation. Pin 16, *INT, is the interrupt that is allowed to work. It is held high, but will become active if forced low by an I/O device. If the pin is probed then it will show pulses are present. Pin 26 is the *RESET. It is held high till the reset is triggered. Then it acts like the 8502.

The last two control pins are 25 and 23, *BUSREQ and *BUSAK. Both pins will show lows when idling. The bus request signal is an input and is active low. It is a request that the Z80 should tristate its address, data and control signals for bus sharing. The Z80 will show a low while the 8502 is in charge, like while idling. The bus acknowledge signal is also active low. It is an output and informs any

device that is taking over the bus lines that the Z80 has indeed tristated. The device can then take over the bus. Pin 6 is the input for the clock. The clock signals are generated in VIC. The Z80 uses a single phase clock. The 8502 uses a dual phase clock. They are discussed in Chapter 17.

The rest of the pins on the Z80 are A15-A0 and I07-I00. They are all three-state active high. The addresses locate registers on the memory map for reading from and writing to. The data pins are the Z80 ports for the two-way data movement.

When the addressing is conducted with an I/O device, the eight lower address bits are able to directly select up to 256 input or output ports. If the refresh register is used, the lower seven address bits address all the rows in RAM.

Z80 INSTRUCTION SET

The 8502 has a Monitor that lets you program or test the chip with machine language. The Monitor is nothing more than a program that acts as an operating system, not unlike the way BASIC in ROM works. Of course there are differences, advantages and disadvantages between the 8502's use of BASIC and the Monitor's machine language using the Instruction Set directly.

The Z80 in the C128 is not so fortunate. There is no Monitor for the Z80 Instruction Set. If you want to program or test the Z80 in machine language, you will have to obtain a Z80 Monitor. There are plenty of them on the market that will do the job on the C128.

If you obtain a Monitor that will let you write Z80 programs, then you'll find that the Z80 has 158 different instructions. They will be listed with the Monitor or assembler that you buy.

The most used instructions in a machine language program deal with the processor reading and writing to the memory map. The map for CP/M is very different than the C128 and C64 maps. They are all discussed in Chapter 16. Probably 70 to 80 percent of the instructions are the read/write variety that move data between the processor and memory. Exchange instructions move data but only swap data between registers.

The next group of instructions have to do with using the ALU. These instructions run data through the ALU which results in arithmetic or logic being executed. The results of the data processing are then placed into the accumulator and the flags relating to the ALU operations are cleared or set accordingly.

One group of instructions shifts or rotates data in the accumulator, in other Z80 registers, or in memory. These instructions have the ability to do binary coded decimal manipulations.

Another group of instructions stops the program counter from routine incrementing and places special addresses in the PC. These are the JUMP, CALL and RETURN instructions.

The I/O instruction group is next. The Z80 is able to address 256 input and 256 output ports. The I/O instructions read or write data between the processor and memory, and memory and external I/O devices.

There are some instructions that are able to halt the Z80 and manipulate interrupts. One instruction, NOP, makes the Z80 have no operation during the time the NOP is processed.

There are some bit-handling instructions. They are able to set, reset, and test bits in some Z80 registers or memory. The results of the instruction execution are then recorded in the flag register.

Lastly, the Z80 is able to transfer any size block of memory to any other group of next-door memory locations. Another instruction lets the Z80 search a block of memory for a desired byte that might be needed.

Z80 CONNECTIONS

The Z80 cannot operate with the 8502 circuits. Specifically it has difficulty with the data bus lines. The address bus is not really a problem, because the Z80 is a three-state device. When tristated, the address lines go into a high-impedance state. Therefore the Z80 address lines can be connected directly to the address bus and can share the bus with the 8502. When there is a possible conflict the Z80 is tristated.

The data lines though are not quite that simple.

Therefore the latch (U12) and the buffer (U13) are used between the Z80 and the data bus. The latch interfaces the Z80 to the data bus during a read and the buffer interfaces the Z80 to the data bus during a write (see Fig. 13-3).

Despite their differences, the 8502 and the Z80 must work together to produce all of the modes. Their handicap is: they cannot use the bus lines at the same time. Although both processors can be on at the same time, they must, however, take turns using the buses.

They do their jobs in the following way. The Z80 is given first use. When you turn on the C128 the Z80 is in charge. The Z80 then performs a lot of initializations of registers. Then, according to how you have started pressing keys or what is in the cartridge port or disk drive, the Z80 will transfer control to the 8502 and the computer will either come up as a C128, a C64, or the Z80 will stay in charge and run CP/M programs.

If a CP/M disk is in the drive, then the Z80 retains control and a CP/M program disk can be put in the drive and installed into RAM. As the CP/M program is run, the Z80 is able to, with the aid of the 8502, use Kernel ROM routines and perform its I/O needs.

The mechanism that allows the 8502 and Z80 to switch control of the bus lines are signals that originate in the MMU, or VIC. These signals pass through some gates and arrive at the Z80 pin 25, *BUSRQ, bus request. The signal from the MMU is called Z80ENABLE, and the signal from VIC is BA. They cause the Z80 to force pin 23 low. That line holds *BUSACK, bus acknowledge. *BUSACK is then sent to the all-important AEC connections through some gates. This makes AEC high and the 8502 RDY line high. The 8502 then takes control. When it is the Z80's turn again, the Z80ENABLE goes low. This makes the *BUSRQ low which generates a high out of *BUSACK. The 8502 then tristates and the Z80 is back in charge.

THE COMPUTER WON'T START

When you look down at the Z80 on the print-board the keyway notch is on the bottom. The 8502,

to the left of the Z80, has its notch at the top of the chip. You could say that the Z80 is mounted upside down. Be careful that you do not get confused as you count pins for test readings.

If the Z80 should conk out, then the first thought is that CP/M won't run but C128 and C64 modes will be okay. Not so! Since the Z80 is so involved with the 8502 and the Z80 is the processor in charge when the machine powers up, a bad Z80 will not let the machine come up.

However, the Z80 is only one of the suspects. Therefore, when a C128 will not start, the Z80 is in the middle of things but not necessarily the seat of the trouble.

The following is a step-by-step procedure to run when the C128 won't start. First of all, check and make sure that the power supply is okay. If there is no power, as outlined in Chapter 24, then the trouble is in the supply; follow the steps in Chapter 24. If the power supply is exonerated, then take a logic probe and begin testing the address and data bus lines of the Z80 and the 8502, with the Test Point Charts of Figs. 12-20 and 13-4.

Pulses should be at all the address and data pins on the 8502. On the Z80, under normal operating conditions, pulses should be on the address pins, but the data pins should be tristating. This is because, while the computer is idling in C64 or C128 mode, the Z80 data pins are inactive. If the machine is in CP/M mode then the Z80 data pins will have pulses.

Should the address or data pins not have prescribed pulses, then you have a valid clue. The next step is to check out the reset circuit. You can check it at either pin 40 of the 8502 or pin 26 of the Z80. Attach a logic probe with the machine off. Then turn it on. The probe should at first go low and then immediately go high. Next press the reset button while the probe is still attached and the computer on. Upon pressing the button the probe should go low. When you release it the probe should then go high again.

Should the above not happen exactly as described you have a number of suspects that could be causing the trouble.

The schematic, Fig. 12-18, shows that the reset circuit is biased around one half of the 556 timer,

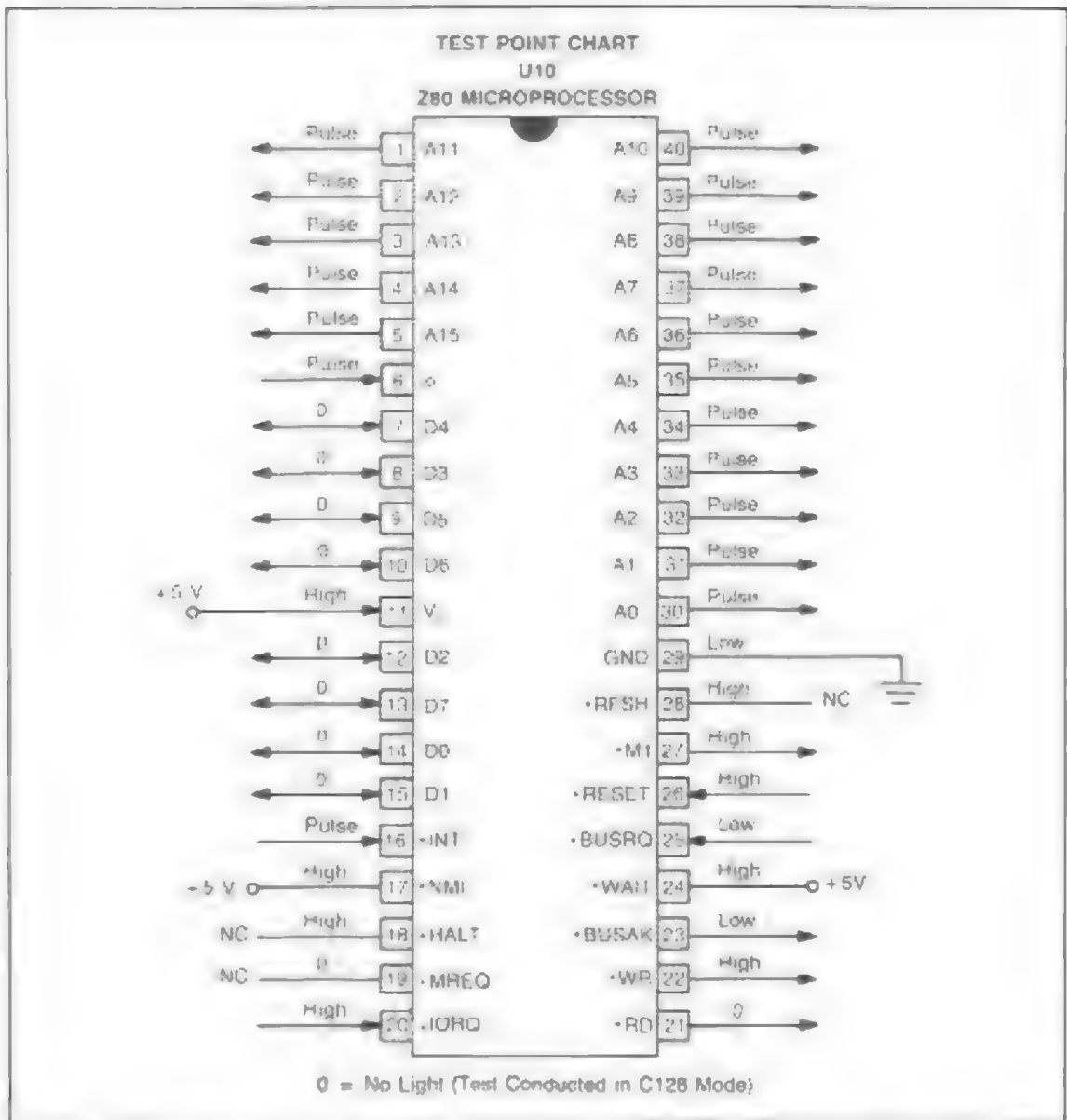


Fig. 13-4. If you find a wrong input logic state, then chances are that the Z80 is okay and the defect is probably in the input circuit to the chip. Should an output logic state be incorrect, odds are good that the Z80 chip itself has troubles.

U27. Also in the reset line is a NOT gate section from U63. Then there is the reset switch, SW2, two 1N914 diodes, CR15 and CR16, two filter capacitors, C91 and C92, and one resistor, R24. These components should be tested one by one. A bad part

could be found and the trouble proven to be in the reset circuit.

Once the reset circuit is checked out as okay, then the next step is to check the clock inputs. At pin 1 of the 8502, a 2 MHz signal is supposed to be

present. At pin 6 of the Z80 a Phase 1 signal is supposed to be present. The logic probe will show both clock signals as pulses. If either or both clocks are missing, then that will explain why the pulses are missing from address and data line pins.

Should the two clocks be present and one or more of the address and data pulses be gone, then the processor exhibiting that behavior is defective. The clock is entering MPU but the MPU is not generating the proper pulsing on the bus pins. The trouble is, most probably, a defect in the MPU internal circuits.

Should the clock inputs not be present, though, the MPUs are probably okay. The trouble is indicated to be in the circuits that are producing the clocks. The 2 MHz clock for the 8502 is coming from

pin 23 of VIC. The Phase 1 clock for the Z80 is coming from pin 25 of VIC. VIC is the generator of all clocks in the C128. Missing clock signals indicate clock circuit troubles (covered in Chapter 17).

The rest of the pins on the Z80 are various forms of controls. The Test Point Chart, Fig. 13-4, shows what logic states should be present and whether the pins are receiving an input from other circuits or are sending signals that the Z80 has processed. Check the pins for wrong logic states. If you find a wrong input reading, then the Z80 is probably okay and the reason for the wrong reading is a fault in the circuits that receive the input. Should there be a wrong output reading, then the Z80 has a defect somewhere in its internals and needs replacement.

14. Programmed Logic Array

U11 is the 8721 PLA chip, shown in Fig. 14-1. It is a 48-pin chip. The PLA for the C128 is stationed right in the middle of the address bus circuits. It works in coordination with the MMU, U7, which is covered in the next chapter. U11 is called a *programmed logic array* but it can't be programmed by you. It has already been programmed, somewhat the way a ROM is programmed, in the factory. It is placed onto the C128 printboard, ready to work, as all the ROMs are.

In the Commodore 64 machine there is also a PLA but it is only a 28-pin DIP. The C128 PLA contains all the workings of the C64 plus more circuits so it can also handle some of the C128 needs. When the C128 is in C128 mode, the PLA and the MMU work side by side. When the C128 is in the C64 mode, though, the PLA does the job without the MMU. The MMU disappears from the memory map during C64 mode. The MMU has register locations on the C128 memory map. The PLA does not have

any memory map addresses. As mentioned, the PLA does not have user programmable registers.

PLA INTERNALS

The PLA is a chip selector. It also performs other jobs which will be discussed later in this chapter. As the name suggests, the PLA works by the manipulation of logic. A PLA is composed of a combination of two arrays of logic: one array of AND gates and another array of OR gates, as Fig. 14-2 illustrates. The AND gates receive the chip inputs and the OR gates deliver the chip outputs.

The AND gates at the chip input pins receive a lot of control signals but the most important inputs are from the address bus. The PLA is connected to lines A15-A10. These are the six highest bits. From these bits and the control signals the PLA is able to produce a lot of outputs.

The AND gates begin the decoding that will result in the chip selects. An AND gate can't output

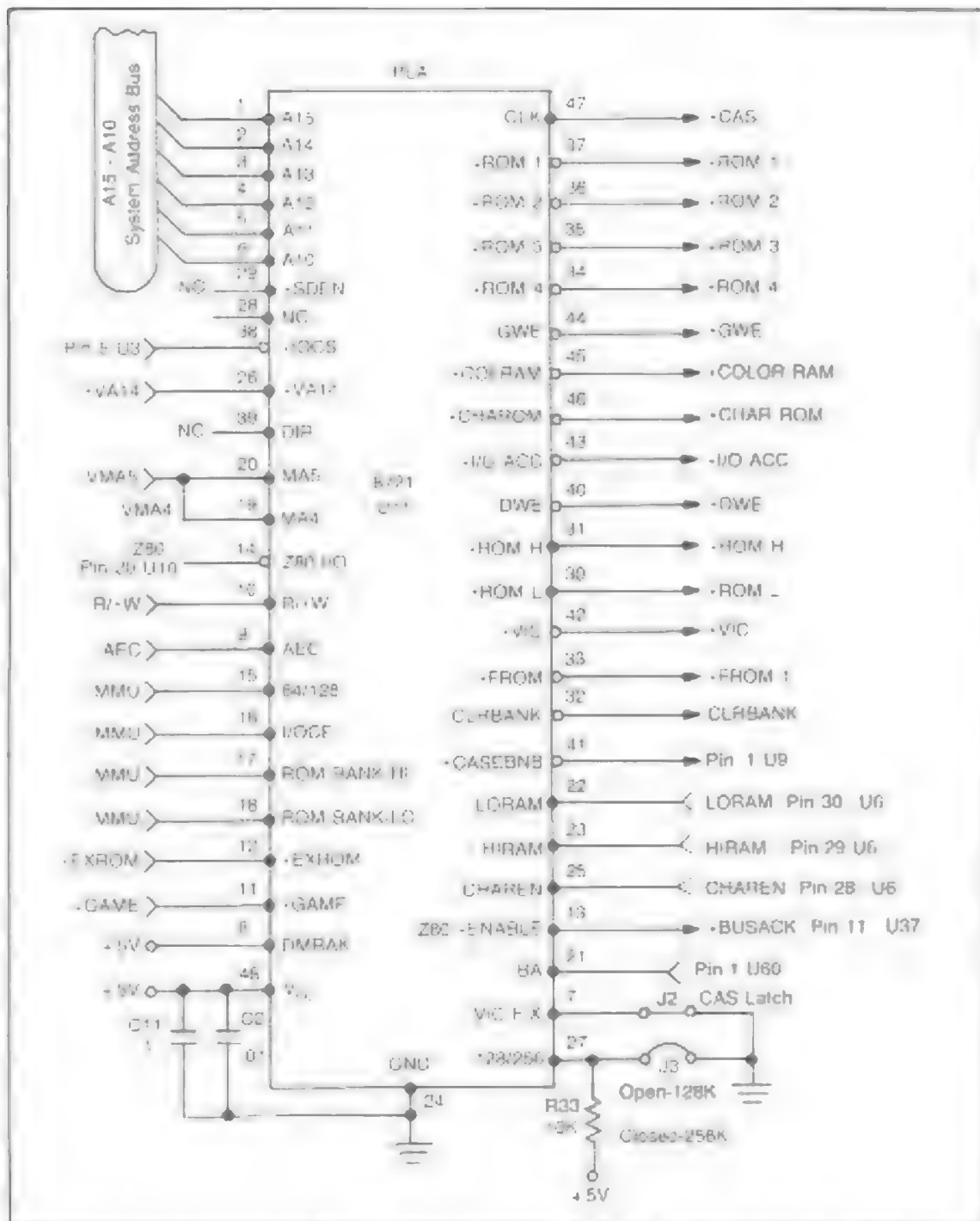


Fig. 14-1 The PLA is stationed right in the system address bus and works as chip selection along with the MMU.

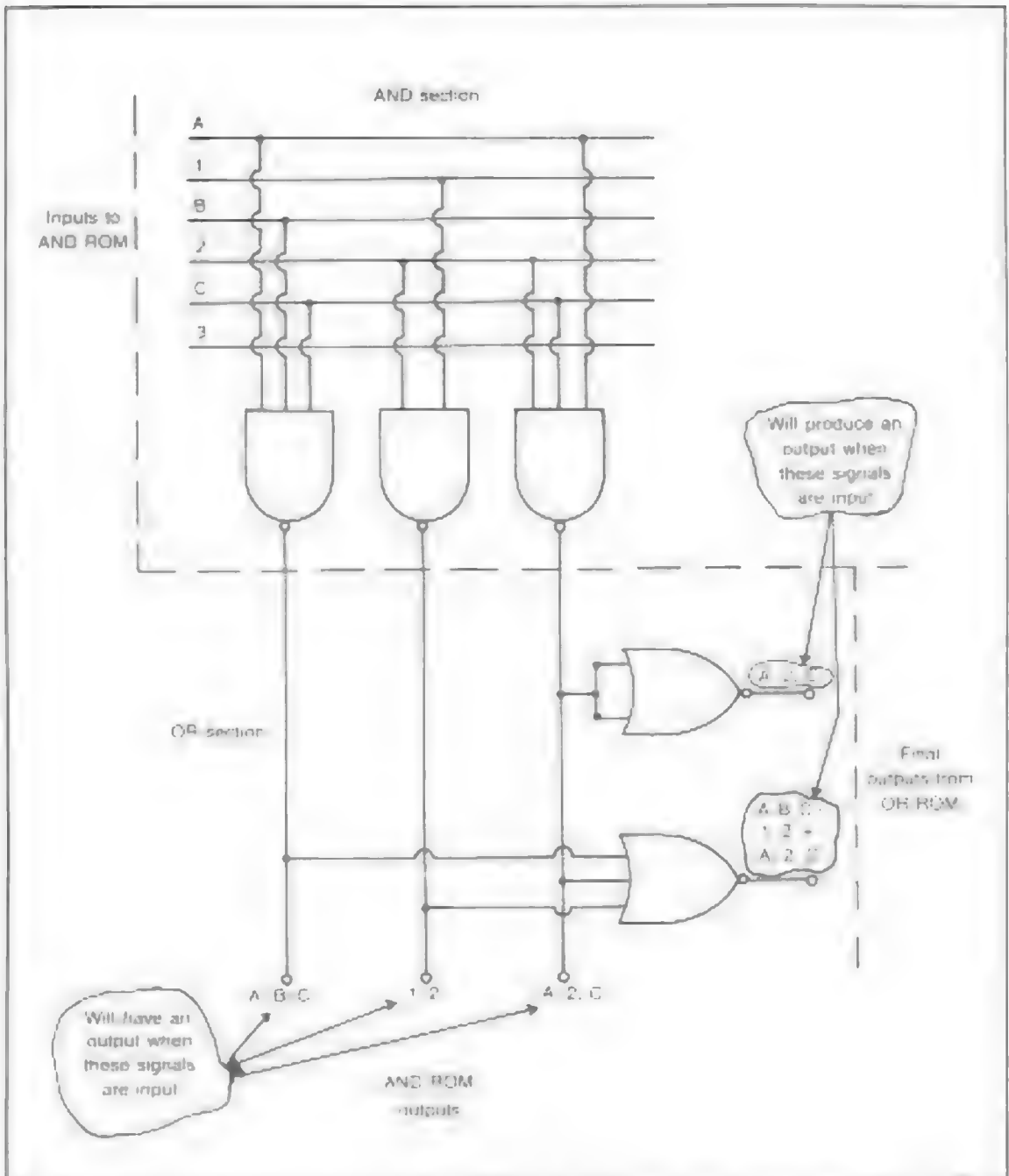


Fig. 14-2 A programmed logic array is a set of AND gates receiving an input and a set of OR gates producing an output. By careful design a particular type of input will predictably always result in a predictable output. With this arrangement, the PLA in the C128 operates as a decoder and produces chip selects and other signals.

a signal, unless all the inputs to the gate itself are highs. The AND gate can have two or more inputs. It only has one output. It will output a high when all inputs are high. The AND gate is an electronic combination lock. It opens up and outputs when all the highs are present. Otherwise it remains off.

The OR gate is so named because it will output a high if any of its inputs are high. The OR gate can have two or more inputs. It only has one output. It will output a high when one or more of its inputs are high.

The OR gate is also an electronic lock. It turns on when a high is inserted into any of its input leads. The difference between the AND gate and the OR gate does not at first appear striking, but it is. Because of this difference in operation, the computer is feasible.

In the PLA, the AND gates are able to distinguish between a lot of different address bit inputs and only output the desired signal to the OR gates. The OR gates are then able to output the correct chip select signal from the designated pin.

The Chip Selects

First of all, the PLA takes care of the C64 mode chip selects, without any help from the MMU. The address bus inputs lines A15-A10 at pins 1-6; as shown in Fig. 14-1. Four control lines and these address bits form a combination that results in one chip select. The four control lines are BA, LORAM, HIRAM and CHAREN.

BA, the bus available signal originates in VIC. It is used to help turn on the RDY in the 8502 besides being inserted into the PLA. LORAM (low RAM), HIRAM (high RAM), and CHAREN (character ROM enable), all come from the 8502. They emerge from the special register pins P0, P1 and P2.

These input signals work their way through the maze of AND and OR circuits in the PLA and output one chip select bit. Figure 14-1 shows the inputs and possible outputs. The chip selected is from the group of I/O, RAM or ROMs that the C64 mode uses.

When a C64 mode cartridge is inserted, the computer detects it and brings into play two more

PLA inputs. They are *EXROM (external ROM) and *GAME. With the additional lines, a number of different memory maps could be turned on, according to the combination of bits that are entering all these inputs.

The C64 memory map is a double-deck affair, shown in Fig. 14-3. The bottom deck is 64K bytes long and is made up of eight 4164 DRAMs. The decking is not one long unbroken stretch. The deck is sectioned off into pages of 256 bytes per page, as shown in Fig. 14-4. This puts 256 pages into 64K of Memory.

On top of the 64K of RAM, using a lot of the same addresses, is 21K more of memory. This gives a total of 85K that the C64 mode uses. Because the 8502 can only address 64K at a time, the PLA is needed to pick and match different memory map layouts for various uses.

Besides the double decking of 85K, the memory map can have some other residents. These are external devices such as the cartridge ROMs. They gain entrance by being plugged into a cartridge holder and activating *EXROM and *GAME.

The C64 mode has a default memory map, Fig. 14-5. It comes up when you start the C128 in C64 mode. Seven other maps are available, besides that default map, by changing bits into LORAM, HIRAM, GAME and EXROM, which change the arrangement of the chips that will reside in the map. Besides that, you may pick one map and then bank in and out of that map the particular chips that are needed. These and other details will be covered in Chapter 16.

In the C128 mode, the PLA also plays a part in the chip select process. It selects all the ROMs, it selects the VIC chip, color RAM and character ROM. It also selects the various I/Os, including the two CIAs and the SID. The PLA does not perform the banking of memory; the MMU is in charge of that.

PLA Additional Functions

Figure 14-1 shows the schematic drawing for the PLA. The inputs and outputs are arrowed. Note the line-up of outputs on the right side of the drawing. You can see all the chip select lines. In addi-

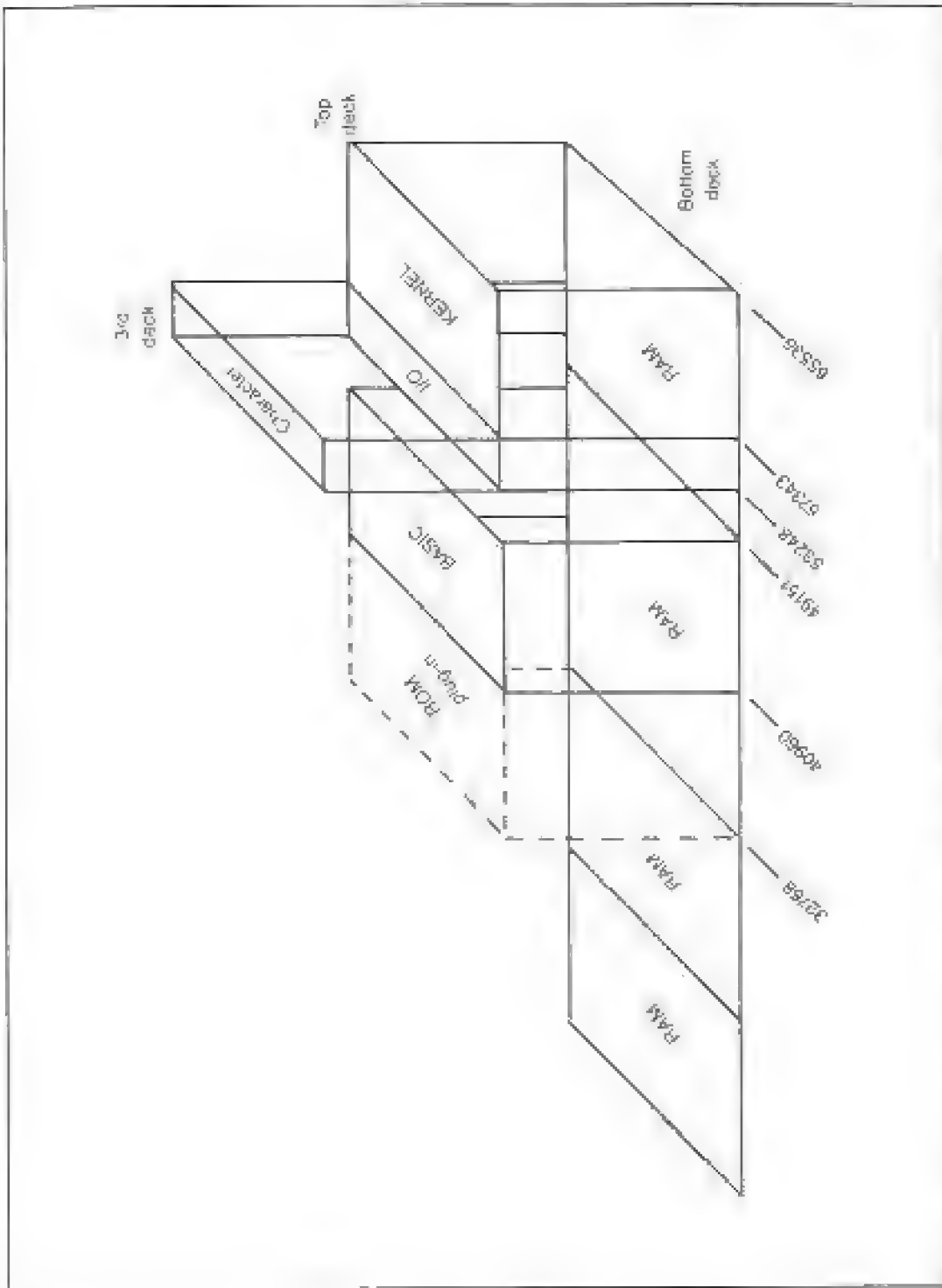


Fig. 14-3 In OS4 mode the memory map is a double decked affair with a small chip set. On the bottom deck is the bit of RAM. The second deck contains the ROMs and IO addresses. The third deck has a single resident, the Character ROM. The PUA picks and chooses chips to make up a 64K layout. There are eight possible layouts in the OS4 mode.

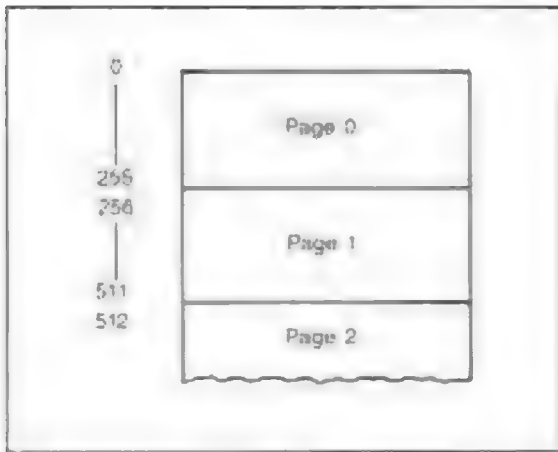


Fig. 14-4 The 64K memory map is partitioned off into 256 pages each containing 256 bytes.

tion, the ϕ CAS (column address strobe) is for DRAMs, and CLRBANK is for the color RAM.

There is ϕ DWE from pin 40 that is the write enable strobe for the DRAMs, and ϕ GWE from pin 44 that goes to U56 (a flip-flop) and then becomes the write enable for the color RAM.

The PLA sends the Z80 signals to turn it on and to handle the Z80's I/O and memory mapping. Pin 13, ϕ BUSACK, is the bit that turns on the Z80. Pin 14, Z80 I/O, goes to pin 20 of the Z80 to get the I/O data moving.

The R/W signal from the 8502 arrives at pin 10 of the PLA. The PLA then takes care of the direction in which the bits on the data bus will be traveling.

Besides all these jobs, the PLA, during the C128 mode, also makes use of the C64 signals: LORAM, HIRAM and CHAREN. These lines in C64 mode are part of the banking scheme. The C64 mode can only utilize 64K at a time. Yet 85K is available between all the chips used for C64. The mentioned control lines aid in banking the chips into eight different configurations for the C64 to use. They are not needed for banking in the C128 mode. The MMU performs the C128 banking duties.

This PLA is able to change the job of these three lines and make good use of them when C128 is in action. The CHAREN input from the 8502 is then used to turn the character ROM off and on in the

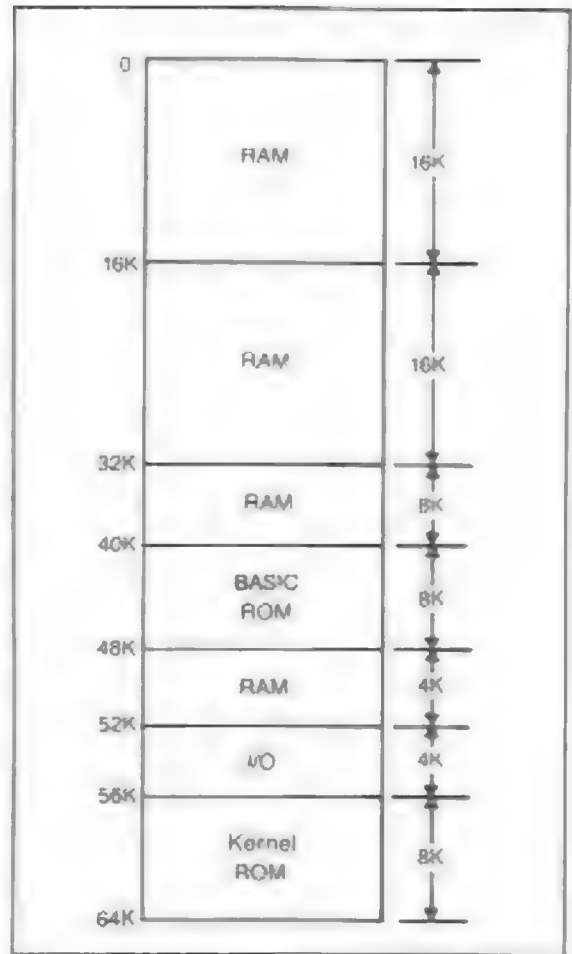


Fig. 14-5. The C64 mode memory default map, the one that comes on when you switch on C64, has the following residents: 40K of RAM, the BASIC ROM, the Kernel ROM and the I/O group.

VIC bank that is selected. CHAREN makes the character ROM readily available in any VIC bank.

LORAM and HIRAM inputs are used to select one of the two sections of the color RAM. Specifically, LORAM input causes the PLA to select one of the two RAM sections when the 8502 is in control. HIRAM input causes the PLA to select one of the two RAM sections while the VIC is in charge of operations. This makes the color displays operate cleanly. It is a valuable feature for programmers and users.

PLA PINOUT

Figure 14-6 is the Test Point Chart for the 8721 PLA chip. Pins 1-6 are the address lines A15-A0. If you touchdown on them with the logic probe, then they will all show activity as the probe reads PULSE. What you are seeing is the current address bits arriving in step with the clock:

Pin 7, called VICFIX, is not being used at present. It is connected to ground and the probe will read a LOW. Pin 8, DMAACK for *direct memory access acknowledge*, is not being used either. It is tied to +5 volts to keep it out of action. The probe will read a high.

Pin 9 is AEC coming from VIC. AEC lets the PLA know when the 8502 is in charge, or when VIC has control. It probes PULSE while the C128 is idling. Pin 10 is the R/W line from the 8502. It lets the PLA know which direction the bits on the data bus should be traveling. Pin 10 shows PULSE.

Pins 11 and 12 are *GAME and *EXROM. They are coming from the expansion bus. When a cartridge is in the expansion port, these signals become active and can affect memory maps. They both probe HIGHS when there is nothing in the expansion port.

Pins 13 and 14 are *Z80 ENABLE and Z80 I/O. They each show PULSE. The enable is a bit from the Z80. When the bit is high the Z80 is in charge. When the bit goes low, the PLA knows that the Z80 is now instating and has given up control. The Z80 I/O is a signal from the PLA to the Z80. It is an interrupt that the Z80 might or might not honor.

Pin 15 shows the computer's mode. If the probe reads HIGH then the machine is in C128. In C64 mode the pin probes a LOW. Pins 15, 16, 17 and 18 all connect directly to MMU pins. Sixteen, a LOW, is an I/O select that reads low. Pins 17 and 18, PULSES, are ROM BANK HI and ROM BANK LO. These four pins 15-18 are the mode status lines. In the C128 mode, between the address bits and these four lines, the C128 banking arrangements are made.

Pins 19 and 20, two more PULSES, are two VIC address bus lines which are multiplexed into pins called MA4 and MA5. These special address lines

help the PLA when the 8502 wants to access VIC or if VIC wants to access the DRAMs. Pin 21 is the BA signal from VIC. It shows a PULSE. Pins 22, 23 and 25 are LORAM, HIRAM and CHAREN from the 8502. Pins 22 and 23 are HIGHS and 25 is a LOW. Pin 24 is also a LOW because it is the chip's GND connection.

Pin 26 is a higher order VIC address line but comes from CIA2. It probes a HIGH. Pin 27 has a jumper in its input line. It sets the PLA for use of 128K or 256K RAM. The jumper is open which sets for 128K RAM. The line is held HIGH. Pins 28 and 29 are not connected. Pin 28 will not show any logic probe readings. Pin 29 will read PULSE from internal connections.

Pins 30, 31, 32 and 33 are four HIGHS showing the signals for ROM L, ROM H, CLRBANK and FROM1. Pins 34, 35, 36 and 37 are the chip select pins for ROM4, ROM3, ROM2 and ROM1. They are all PULSES except 37 which is HIGH.

Pin 38 is an I/O chip select that is coming from the U3 decoder. It reads PULSE. Pin 39 is not connected but will read a pulse as signals spill over inside the PLA and make it to the pin. The PULSE has no practical meaning. Pin 40 is the *DWE output, a write enable to the DRAMs. Pin 41 is a PULSE that goes to a pair of NAND gates in U9. The pin is called *CASENB for column address strobe enable B. After passage through the two NANDs it becomes *GCAS0 and *GCAS1. These two signals are NANDed again and are then applied to the two banks of RAM as *RAMCAS0 and *RAMCAS1. Each strobes its respective RAM banks for the column addresses.

Pin 42 is the *VIC chip select. It reads a PULSE. Pin 43 is the *I/O ACC and is applied to VIC as a PULSE. Another PULSE from pin 44, *GWE, is the write enable that ends up at the color RAM. Pin 45 output, another PULSE, is the *COLOR RAM chip select. Pin 46, still another PULSE (called *CHAR ROM), is the chip select for the character ROM.

Pin 47, *CAS, is the companion PULSE that is NANDed with GCAS0 and GCAS1 gates to form RAMCAS0 and RAMCAS1, which are applied to

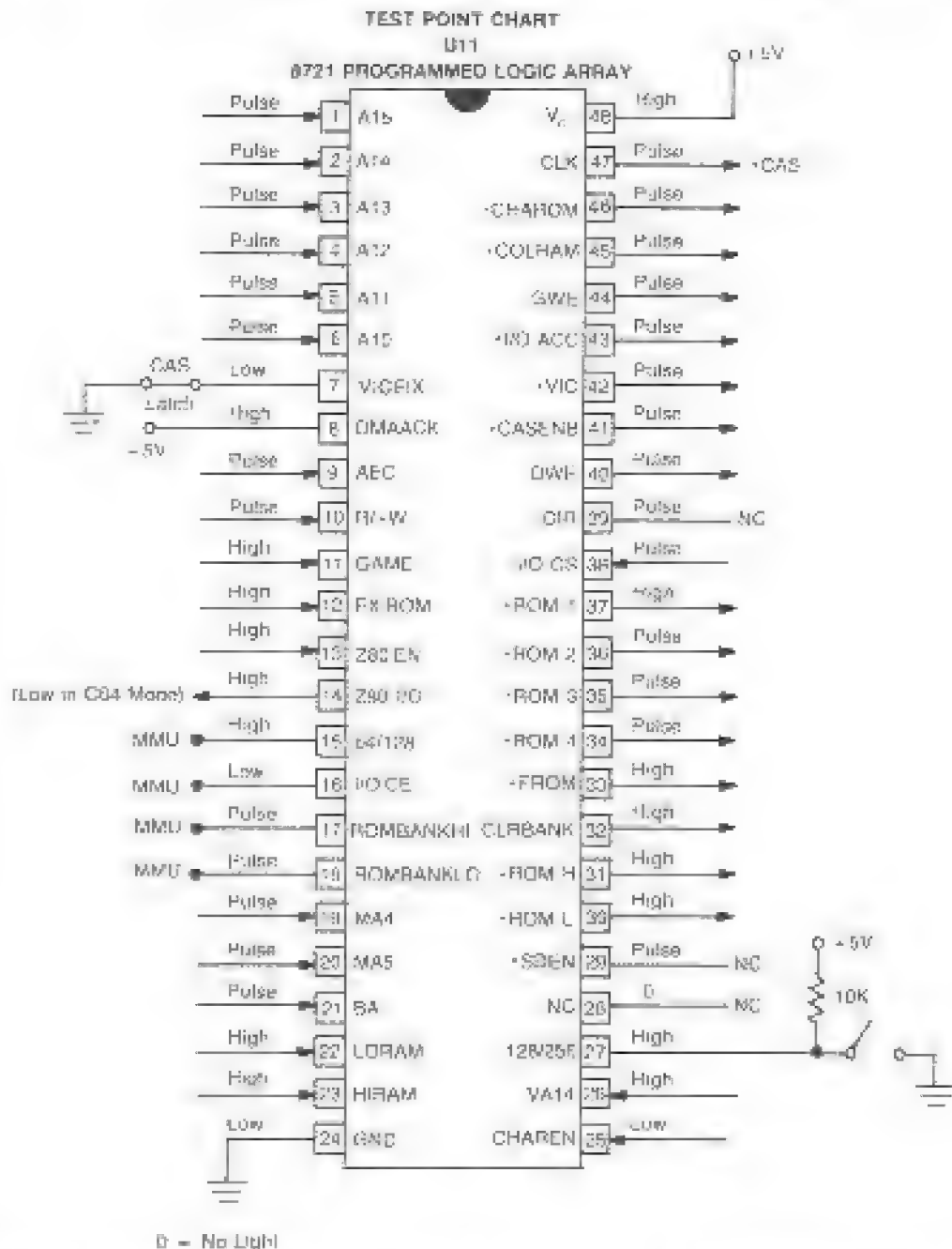


Fig. 14-6 The PLA provides all the chip selects for the C84 mode. In C128 mode though, it just becomes a helpmate for the MMU.

RAM banks 0 and 1. Pin 48 shows a HIGH. It is the +5 volt power input pin.

To sum up, the 48-pin Programmed Logic Array is the only large chip in the C128 that cannot be programmed. It has already been built with a massive switching program installed. It receives inputs from address lines, control lines from the 8502, the expansion port, VIC, CIA2 and a few other places. It interprets these lines and then outputs chip selects

and other decoded signals.

The PLA provides all the chip selects and other necessary signals for the C64 mode. For the C128 mode it does a lot, but not everything. The MMU does most of the interpreting for the C128 mode using the PLA as a helpmate. The next chapter on the MMU goes into more detail on how the PLA operates with the MMU.

15. Memory Management Unit

The last chapter showed how this computer in C64 mode uses the PLA and its input signals to configure 85K of available memory into eight useful maps of 64K each. A maximum of 64K is in each map because the 8502 is only able to directly address 64K with its 16 address lines.

In the C128 mode and the machine language Monitor mode, 372K of physical memory is available that the 8502 must be able to address. The 8502 still can only address 64K at a time. That is why the Memory Management Unit, U17, is in the C128 design. This MMU, with the help of the PLA, configures 16 different bank formats, and has the 8502 address all 372K of memory by quickly switching from bank to bank.

In each bank are carefully selected amounts of RAM, ROM, UC and other addresses. The amounts of each type of memory have been decided upon in the factory as the ones that would be most useful in most applications. As a program is run, the program specifies different banks at different places in the routine. A bank is chosen by typing in BASIC,

the command **BANK** and then a number between 0 and 15.

Table 15-1 shows the arrangement of the banks. Note that banks 0-3 are four 64K stretches of RAM. Does this mean that the MMU is capable of addressing 256K of RAM? Yes! However, there is only 128K of RAM in the C128. These are extra capabilities that will allow upgrades of the C128 in the future. Such a new machine would be the C256. Furthermore, if 256K DRAMs were used instead of 64K DRAMs, a C1024 would be created.

Anyway, lets stay with the C128 for the present. The additional 128K that can be addressed would be in banks 2 and 3. If you choose bank 2 then you will be addressing bank 0. Should you dial up bank 3, you will be addressing bank 1.

BANK CONTENTS

Table 15-2 is a listing of what forms of memory are in the banks and their addresses in decimal and hex. The banks are numbered 0-15 in decimal and in hex are 0-F. The decimal is needed when testing

Table 15-1. In the C128 mode, the MMU can get the computer to operate with 16 different memory map layouts. In BASIC the banks are called out using decimal numbers. In the Machine Language Monitor the banks need hexadecimal to appear

Bank Number		Bank Contents
Decimal	Hex	
0	0	RAM Set 0
1	1	RAM Set 1
2	2	RAM Set 2 (not in C128, uses RAM Set 0)
3	3	RAM Set 3 (not in C128, uses RAM Set 1)
4	4	RAM Set 0, IO, Functional ROM (empty socket)
5	5	RAM Set 1, IO, Functional ROM (empty socket)
6	6	RAM Set 2, IO, Functional ROM (empty socket)
7	7	RAM Set 3, IO, Functional ROM (empty socket)
8	8	RAM Set 0, IO, Cartridge ROM
9	9	RAM Set 1, IO, Cartridge ROM
10	A	RAM Set 2, IO, Cartridge ROM
11	B	RAM Set 3, IO, Cartridge ROM
12	C	RAM Set 0, IO, Kernel ROM, Internal ROM Low
13	D	RAM Set 0, IO, Kernel ROM, Cartridge ROM Low
14	E	RAM Set 0, Kernel ROM, BASIC ROM, Character ROM
15	F	RAM Set 0, IO, Kernel ROM, BASIC ROM

Table 15-2. There are fourteen possible residents the MMU can pick and choose from. RAM sets 2 and 3 are not in the machine. If you choose them, then RAM sets 0 and 1 will switch in instead.

Bank Residents	Addresses	
Type of Memory	Decimal	Hex
RAM Set 0	0-65,535	\$0000-\$FFFF
RAM Set 1	0-65,535	\$0000-\$FFFF
RAM Set 2 (same as 0)	0-65,535	\$0000-\$FFFF
RAM Set 3 (same as 1)	0-65,535	\$0000-\$FFFF
{ Functional ROM U36	32,768-53,247	\$8000-\$CFFF
{ Functional ROM U3	57,344-65,535	\$E000-\$FFFF
{ IO	53,248-57,343	\$D000-\$DFFF
{ Cartridge ROM	32,768-53,247	\$8000-\$CFFF
{ Kernel ROM	49,152-53,247	\$C700-\$CFFF
{ Kernel ROM	57,344-65,535	\$E000-\$FFFF
{ BASIC ROM	16,384-49,151	\$4000-\$BFFF
{ Character ROM	53,248-57,343	\$D000-\$DFFF
{ Memory Management Unit	65,280-65,284	\$FF00-\$FF04
{ Memory Management Unit	64,528-64,533	\$D501-\$D505

in BASIC and the hex is used when testing in the Monitor. Note that addresses start with zero in decimal and \$0000 in hex. The \$ sign means hex. Each bank ends up with 65535 in decimal and \$FFFF in hex. The banks are 64K segments.

The first four banks 0-3, in Table 15-1, are all RAM except for some control addresses that are needed. The Memory Management Unit is involved with every bank. The MMU has addresses 65280, 65281, 65282, 65283 and 65284. These addresses

in every bank are reserved for the MMU. Furthermore, every bank starts off its layout with some RAM from the 64K block 0. The amounts of RAM from this group of chips varies. Banks 1, 2, 3, 5, 6, 7, 9, 10 and 11 use 1024 bytes from RAM block 0. Banks 4, 8, 12 and 13 use 32768 bytes of block 0. Banks 14 and 15 use 16384 bytes of 0 RAM.

The MMU registers appear in every bank for the C128 to be able to switch from bank to bank so easily. If the MMU registers did not have residency in every bank, then once you got into a bank there would be no way out. Chapter 16 has more details about this.

During power up, the ROMs place copies of the instructions to handle IRQs, NMIs and resets into sections of the RAM that will be used. That way, even if you are in a bank that does not include the ROMs, when these interrupts happen, the program won't crash—the interrupt will be serviced properly.

Banks 4-7 are there to work with the empty ROM socket, called U36. The banks have various amounts of RAM from blocks 0-3 in the bottom 32K of the 64K bank. In the top 32K of these banks, the Functional ROM, U36, is installed. It resides at addresses 32768-53247 and 57344-65280. There are ROM chips appearing on the market, like the Superchip mentioned earlier. If you plug one of these internal function ROMs into socket U36, you can then use banks 4-7. Otherwise you have no need for these banks.

Banks 8-11 are like the 4-7 banks except that they handle a C128 External Function Cartridge ROM. The cartridge ROM plugs into the expansion port. The ROMs use the same addresses as the Internal Function ROMs, even though they are not physically in the same place. There is no conflict even though they use the same addresses because the difference in bank numbers keep them separate. There is also no conflict with C64 cartridges because the machine can only come on in C64 mode when such a cartridge is in the expansion port.

Bank 12 also is only used when an internal function ROM is plugged into the U36 socket. The Kernel ROM is included in this bank, however. Bank 13 has the same configuration except that it deals with the external function ROM in the expansion port.

Banks 14 and 15 are the ones that are often used. Both contain RAM from the 0 block at locations 0-16383. The BASIC 7.0 ROM is in both at addresses 16384-49151. The Kernel ROM is in both banks at 49152-53247.

In bank 14 at 53248-57343 is the Character ROM. At these same addresses in bank 15 is the I/O system, sound, video and other I/O chips. The Kernel ROM also has addresses at 57344-65535 in both banks 14 and 15.

Remembering all the contents of the 16 banks can be broken down into two groups. One group has the banks used without the internal and external function ROMs. Banks 14 and 15 are the ones with all the ROMs and I/O. Banks 0 and 1 are the banks with practically all RAM. Banks 2 and 3 are just copies of 0 and 1. As long as you do not use a functional ROM chip in U36 or plug a C128 cartridge in the expansion port, just inform yourself on these banks.

When you turn on the machine in C128 mode you come up in bank 15. For routine BASIC 7.0 programming, you will work in bank 15 most of the time. The BASIC commands though are simply shortcuts that in turn are interpreted and call out machine language programs that are the actual computing. The BASIC commands then cause the machine to switch from bank to bank as the program lines are executed. After the lines are run off, the machine returns to bank 15, unless you issue a BANK command.

In the Monitor though, the rules are changed. When you enter the monitor, bank 0 is automatically brought on, bank 0 is mostly RAM. If you try to go directly to bank 15 to access ROM routines, it won't work. In order to switch banks you must write to the MMU. This will be discussed later in this chapter.

MMU REGISTERS

Two sets of registers are in the MMU. The five registers mentioned earlier appear in all 16 banks at addresses 65280-65284. These registers are four load configuration registers (A, B, C and D), and the main configuration register. These registers are al-

ways addressable in any bank as long as the C128 mode is being used.

Twelve more registers are at addresses 54528-54539. These registers are grouped with the I/O system addresses in bank 15. If the bank should be changed, then this group of MMU registers will disappear from the view of the 8502. That is the reason for the other group of five MMU registers that are in every bank.

As seen in Fig. 15-1, the register that appears in the always-present group and in the bank-15-only group, is the configuration register, the CR.

In BASIC, all of these registers are configured automatically by the routines in the ROMs. In machine language though, the programmer must configure them by writing to them with the 8502. There are four load configuration registers, LCRs, four pre-configuration registers, four page pointer registers, one mode configuration register, one RAM configuration register and a version register besides the configuration register that appears twice, at 65,280 and 54,528.

The configuration register is the most important register in the MMU. Writing to the CR will set up any bank that is desired. The 16 banks are 16 different settings of the CR. Let's see exactly what that means.

Setting the CR

Figure 15-2 shows the byte of bits in the configuration register. Writing to this register sets five groups of bits. Bits 7 and 6 define the amount of RAM that will be placed into the desired bank. Two bits are able to form four bit combinations. If you place binary 00 into bit positions 7 and 6, then the circuits will choose from block 0 of RAM. When bits 01 are installed, you have arranged to choose RAM from block 1.

If 10 is placed into positions 7 and 6, then you have chosen from the not-available block 2. In the C128, at present, you'll simply be switched back to block 0. Should you place binary 11 in the bit positions, you'll be switched back to block 1 because there is no block 3. The MMU is able to choose

blocks 2 and 3, providing 256K, but the chips are not in the machine.

To sum up: by writing bits to the CR bit positions 7 and 6, you can choose the RAM that you want in the bank you are going to work in.

Bit positions 5 and 4 are also capable of having four choices by the careful placing of bits. The bits 00 will have the internal MMU circuitry choose the Kernel ROM for use in the bank to be configured. Bits 01 choose any internal function ROM that might be plugged into U36. The binary bits 10 do the same thing for any external function C128 ROM that might be plugged into the expansion port. Finally, bits 11 will pick RAM to be in the bank.

Bit positions 3 and 2 in the CR follow the same formula. Bits 00 are the ones that will get the BASIC ROM into the desired bank. Bits 01 again choose the internal function ROM and bits 10 the external function ROM. Bits 11 choose from RAM.

Bit position 1 can only have two states. Therefore, it can only choose one of two memory components to be in the bank which the CR byte is forming. When position 1 is a 0, the circuits install the BASIC ROM low for the bank. If a 1 is installed in bit position 1, then RAM will be in the bank.

Bit position 0 also can assume only two states. When it is a 0, the I/O system is placed in the bank. If a 1 is placed in the register bit, then the Character ROM, or as an alternative some RAM, could be chosen.

As an example, let's set up bank 14. Bank 14 calls for: RAM block 0; the Kernel ROM; the BASIC ROM; and the Character ROM. To place RAM 0 into the bank, bit positions 7 and 6 receive bits 0 and 0. Positions 5 and 4 receive 0 and 0 too, and the Kernel becomes a member of the bank. Positions 3 and 2 receive 0 and 0 also to have the BASIC ROM high. Position 1 also gets a 0 for the inclusion of BASIC ROM low. Lastly position 0 gets a 1 for the Character ROM to be included in the bank. The total CR byte becomes 00000001 for bank 14 to be seen by the 8502.

Bank 15, the one that comes up as you turn on the C128 mode, is almost identical to bank 14 ex-

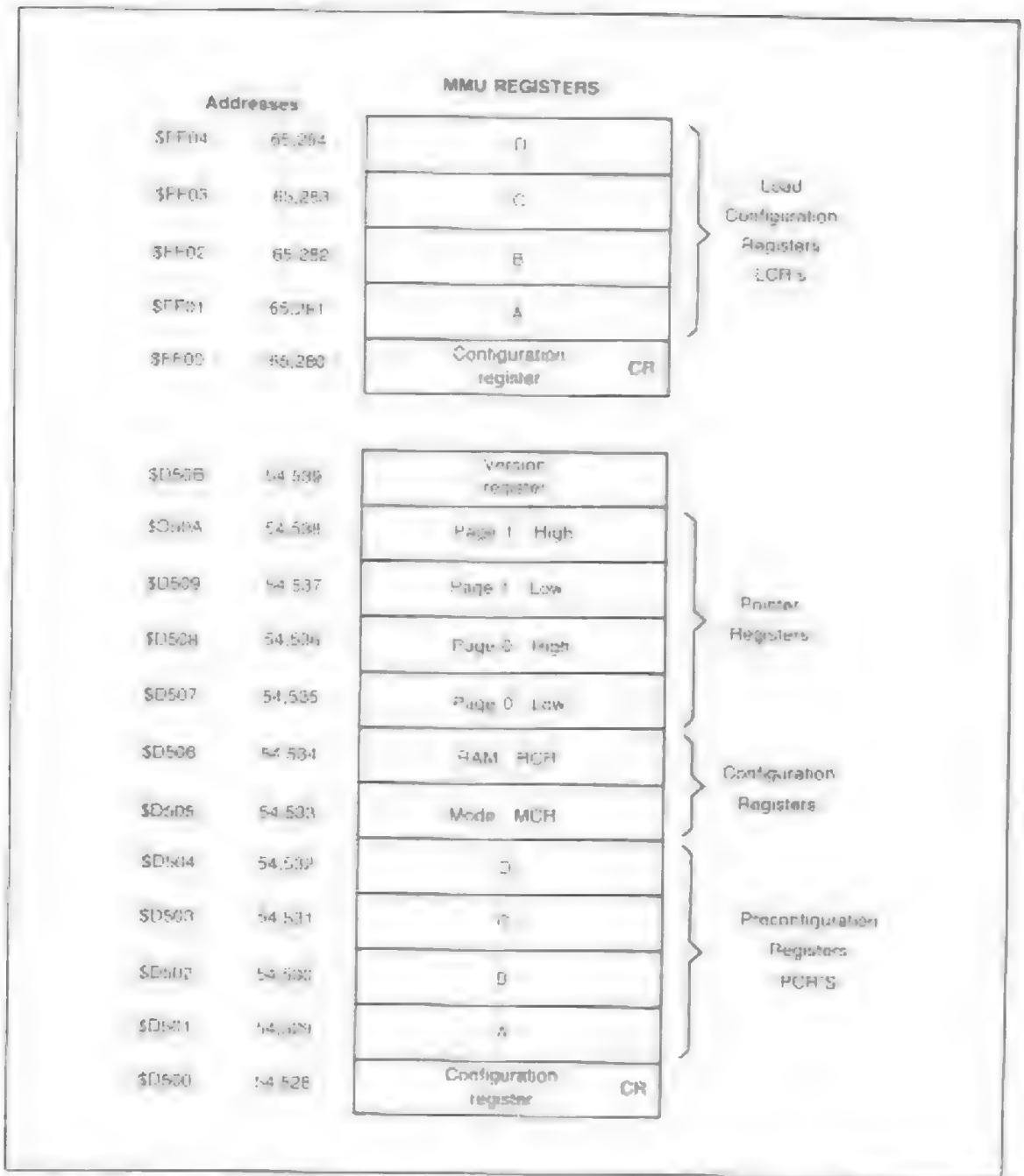


Fig. 15-1 There are two sets of registers in the MMU. The ones at 65,280-65,264 are installed in every one of the 16 banks. The most important register, the CR, is repeated in both register sets at addresses 65,200 and 54,528.

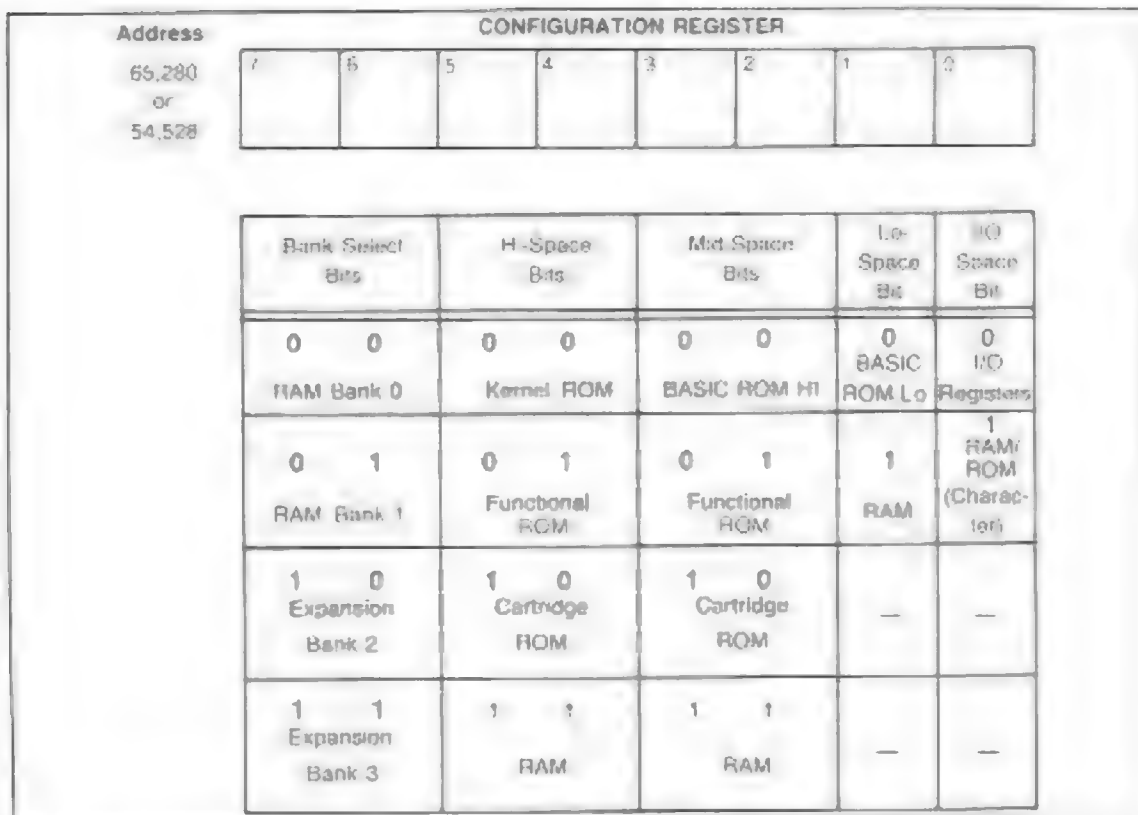


Fig. 15-2. The programmer can configure five sets of bits by writing to the CR.

cept for bit position 0. In bank 15, position 0 is 0. That makes the CR byte (00000000). The only difference between bank 14 and 15 is bank 15 chooses the I/O system on the memory map and bank 14 chooses the Character ROM instead.

The CR is found with two addresses: one is at 54528 and the other at 65280. The register at 65280 is accessible to the 8502 no matter what bank is being used. The location at 54528 is not always available in some banks. That is why the CR is given two addresses. Without access to the CR you can't always switch from bank to bank.

Setting the Preconfiguration Registers

It is easy in BASIC to switch from bank to bank. You simply insert a program line like:

```
220 Bank 14
```

into the BASIC program. The 8502 is then instructed by the BASIC ROM to place binary bits 00000001 into the configuration register.

In machine language it takes a little more work. You have to perform a write operation and install the bits in the CR. That isn't too difficult either. Then there are ways to change banks by using other registers in the MMU. This becomes valuable to the programmer. It could be useful to a troubleshooter who wants to exercise the MMU to see if it is performing properly.

In the set of MMU registers at 54528-54539 there are four preconfiguration registers as shown in Fig. 15-1. These are known as PCRs. In the group of MMU registers at 65280-65284, there are four corresponding registers called load configuration registers, LCRs. The PCRs and LCRs are each called A, B, C and D respectively. The A's are a

working pair, the H's are a working pair, and so on. Each pair together can be preset to change the set of bits in the configuration register, which switches banks. Then a single STORE instruction will trigger the LCR to make the PCR place its contents into the CR and thus switch banks.

It works like the following. In the beginning of a program you can store up to four bank changes in the PCRs. Then at appropriate places in the program there will be STORE instructions to the appropriate LCRs. It should be noted that the PCRs are not always available to the 8502 in every bank. It doesn't matter though because the LCRs are always available in every bank. All that has to be done is write a STORE instruction to the available LCR and it will trigger its companion PCR even though the 8502 can't access the PCR. The LCR can always get through. Incidentally, the STORE instruction to the LCR can have any value attached. It is the STORE instruction that is the trigger, not the data contents of the program line. When the STORE instruction is executed to the LCR, the PCR is triggered and replaces the contents of the CR, with the contents the LCR had been preconfigured with. The CR then causes the current bank to be switched in accordance with the new contents of the configuration register.

The above is what is going on in the CR, PCRs and LCRs. If you are using BASIC, then you are not supposed to do anything with these registers. BASIC will handle them all. If you do alter the register contents then BASIC could become very confused and anything could happen including a program crash.

You can look into the MMU registers and see their contents in machine language.

Inspecting the MMU Registers

Because the MMU has 17 registers with addresses, it is easy to inspect them and make sure they are acting normally. The easiest way to look over the registers is with the Monitor. Of course you can only inspect the registers in this way if the C128 is operating. Should the computer be down completely, the only troubleshooting measures you

can take are with test equipment like the logic probe and vom.

The MMU register inspection begins by entering the Monitor. When you enter, either by commanding MONITOR or pressing F8, you'll see the sign-on:

```
MONITOR
PC      SR AC XR YR SP
; F8000 00 00 00 00 F8
```

The Monitor opens with a hex read out of its programmable registers. The contents of the PC are shown as F8000. Because each hex number represents four binary bits, F8000 would appear to be 20 bits. There are only 16 bits in the program counter. The F is not in the PC. It shows what bank the 8502 sees at this time. F in hex is 15. The 8502 is looking at bank 15. The rest of the registers all have two hex numbers representing eight bits, as they should.

When your Monitor signs on in this way, it shows that a lot of the circuits in the computer are working fine. All the circuits that are instrumental in placing the Monitor sign-on into the TV display are doing so. If the sign-on was incorrect, that could indicate troubles.

Getting back to the MMU registers: the Monitor has two commands that are valuable for troubleshooting. One is M for Memory. M is somewhat like a PEEK in BASIC; M followed by a start address and an end address, both in hex, will display those addresses included from the start to the end. For example, if you enter M FD500 FD50B, then you will display:

```
> FD500 00 3F 7F 01 41 B7 04 00
> FD508 F0 01 F0 20 FF FF FF FF
```

What does all that mean? You have just displayed the contents of bank 15, locations D500 through D50F. These 16 locations include the MMU registers D500 through D50B. The extra four registers shown, from D50C through D50F, are thrown in for good measure because the M command displays a minimum of eight registers on a line. The

Monitor starts each line with the first address shown, and from then on in multiples of eight. Note that the first line is address D500 and the second line begins with D508.

The four preconfiguration registers are in this group of MMU registers: The PCR's will always display the same contents as their respective LCR's have. You'll see that when you display the set of registers next that contain the LCR's.

If you are wondering what those symbols are on the right of each line: they are the ASCII characters corresponding to the eight memory locations on the line.

Once you have examined the twelve MMU registers in that group, then you can look at the remaining five in the other group. Enter:

M FF00 FF04

When you press RETURN the display will show:

```
>0FF00 3F 3F 7F 01 41 78 48 8A
```

The display shows MMU registers, FF00-FF04, plus the contents of registers FF05, FF06 and FF07. After the hex contents are the ASCII symbols that are called by the contents of the eight registers. When this display comes up, these MMU registers are indicated to be okay and operating normally. Note the 0 in front of the address that appears. That shows the bank that the monitor is using at this time. It is bank 0 which is mostly RAM except for the upper MMU register and some control signals.

The other Monitor command that lets you inspect the MMU registers is D. It is the disassemble command and is used by the programmer to list programs in memory. It can be used to inspect the MMU registers. Just enter the first address of the group of registers:

D D500

The display will bring up a page of memory—in this case addresses D500-D514. The contents of

these addresses will be shown. The contents are listed vertically alongside the address. If you enter:

D FF00

then the same type of display will be shown. Other information is in the display too. For example, the period at the beginning denotes the disassembly. The 0 in front of the hex addresses is again the bank number. There are also machine language symbols present. This is disassembly information, not necessarily having anything to do with the inspection test of the MMU registers.

MODE CONFIGURATION REGISTER

Address D505 is the MCR (see Fig. 15-3). It makes two vital decisions. First it decides which processor should be in charge, the 8502 or the Z80. Secondly, if the 8502 is in charge, then it decides what mode should be used: C128 or C64. The MCR makes these decisions by the type of bits that are installed in the register. Each bit activates a particular set of circuits in the MMU that switches in the desired processor and the mode of the 8502. If CP/M is to be used, then the Z80 will be given control.

Bit position 0 chooses the processor. If the position has a 0 then the Z80 will be in charge. Should a 1 be placed in that position, then the 8502 will get the controls. When the C128 is first powered up, the bit position is at 0. This lets the Z80 get the motor started and the machine initialized. As soon as the Z80 does its work, the position state is changed to a 1 and the 8502 takes over.

Should a CP/M disk be in the disk drive, the bit remains at the 0 state and the Z80 retains control of the computer. The Z80 then loads the disk contents into memory and the computer is ready to be programmed as a Z80 computer or be loaded with CP/M programs to do some useful work. The CP/M operating system sections in ROM are accessed.

Bits 1 and 2 in the MCR are not used. They are there as extra capacity. In the future they could be used in case additional modes are installed in the computer system. If you recall, the MMU is also

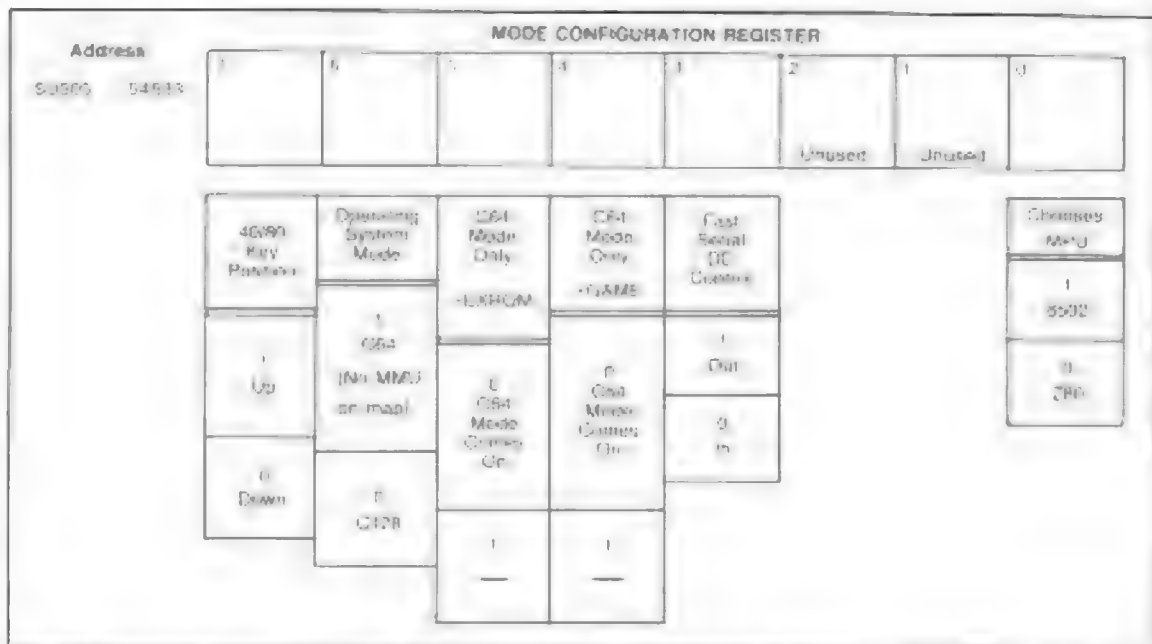


Fig. 16-3 The MCR makes two decisions: it decides which processor is to be in charge and, if it picks the 8502, which mode should be used: C64 or C128.

able to address another 128K of RAM if RAM blocks 2 and 3 are ever installed in addition to the present RAM blocks 0 and 1. Bits 1 and 2 of the MCR are also additional power the MMU can bring to bear if ever desired.

Bit 3 is a FAST serial input and output control. If it is a 0 then it sets an input for FAST serial input from the disk drive. If set at 1 it causes an output operation from the serial bus buffer. In the present day C128, it is not used as an input.

Bits 4 and 5 are the *GAME and *EXROM sense bits. When a cartridge is in the expansion port these lines sense it and force the C128 into a C64 mode. The cartridge then takes over control and runs the computer.

The C64 cartridge will make one of the *GAME or *EXROM lines go low. When that happens either bit 4 or 5 will have a 0 state and the C64 mode will be actuated. If a C128 cartridge is in the expansion port, then these two lines will not be affected and both bits 4 and 5 will have a 1. The computer will stay in C128 mode.

Bit 6 also is a C64/C128 mode setter. It selects the operating system. If the bit is low then all of the MMU registers are turned on and the machine goes into C128 mode. Should the bit go high, then the C64 mode takes over.

Bit 7 is affected by the up or down position of the 40/80 DISPLAY key. When the key is in the up position, bit 7 is held high and the computer will display 40 columns. The video output will exit from the RF Modulator or the composite video port. Should the key be pressed into the down position, then bit 7 goes low and the computer will output 80-column video from the RGB output port.

The contents of the mode configuration register can be inspected quickly by entering the Monitor and typing M FD505. A display will appear showing 48 registers from D505 on. Ignore them all except for D505. Upon start up it will read hex B7. Translated to binary B7 is 10110111. Counting from the left, the bit positions are 7, 6, 5, 4, 3, 2, 1.

Bit position 7 has a 1. That selects the 40-column display output. Bit position 6 has a 0. That

puts the mode into C128. Bit positions 5 and 4 are both high. That says there are no C64 cartridges in place and the C128 mode holds intact.

Bit position 3 is a 0. No input is in this machine so ignore it. Bits 1 and 2 are held high but they are unused and to be ignored too. Bit 0 is a high. That means the 8502 is the chosen processor. That is how the MCR conducts its business as the C128 first comes on normally

RAM CONFIGURATION REGISTER

The hex address of the RCR is D506. The RCR controls the amount of common RAM that is shared between the two 64K blocks, 0 and 1. It also de-

terminates how the common RAM is to be shared. Lastly it designates the RAM block that the VIC can use.

Figure 15-4 shows the four ways that the shared RAM is assigned. There can be 1K, 4K, 8K or 16K amounts shared. Bits 0 and 1 of the RCR, dictate the amount of RAM that is to be shared. There are four possible combinations of the two bits.

Figure 15-5 shows the amounts of shared RAM in accordance with the states in the bit positions. In Fig. 15-5 the bit positions 2 and 3 show where the shared RAM is to be placed. Bits 4 and 5 are unused and reserved for future use. This is another example of the power this MMU has and indicates future potential of C128 computers.

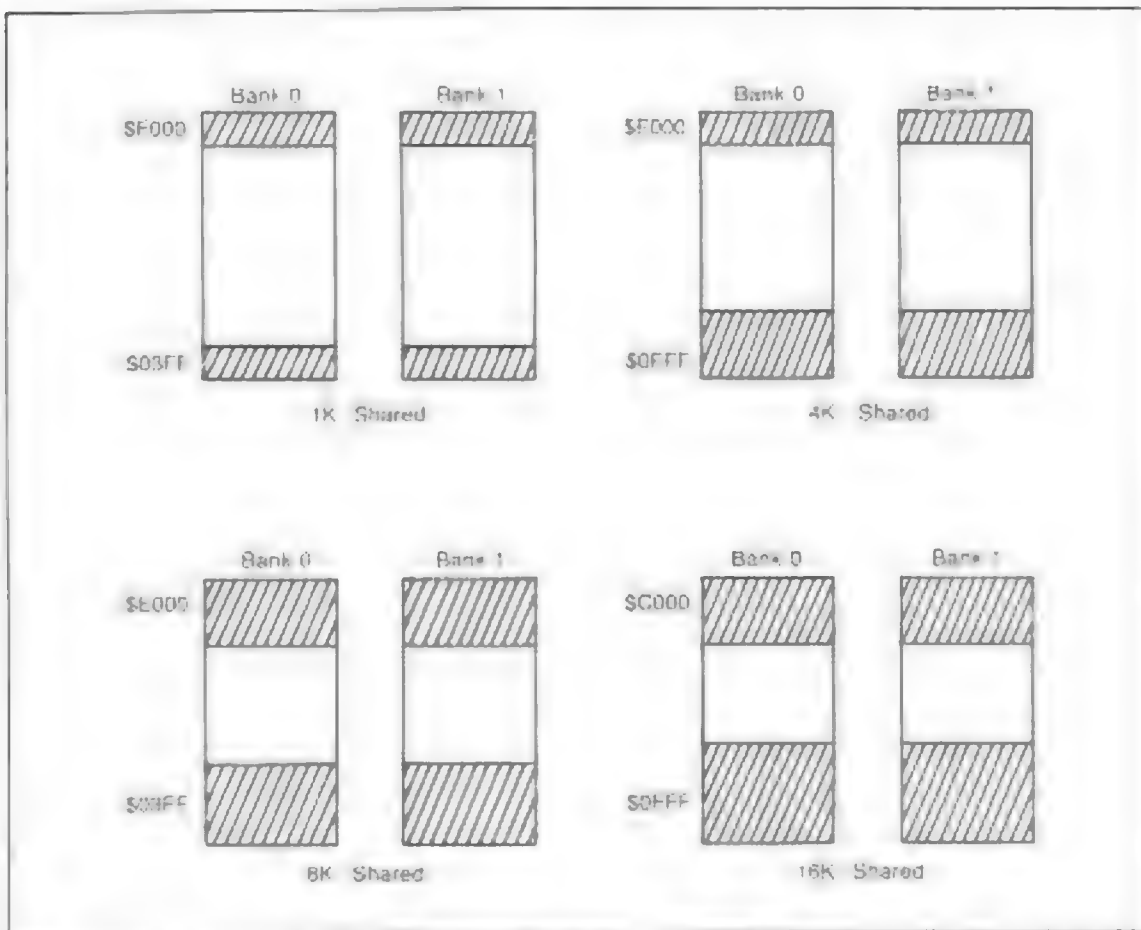


Fig. 15-4 The RAM configuration register picks the amount of common RAM that is to be shared between the two 64K blocks 0 and 1.

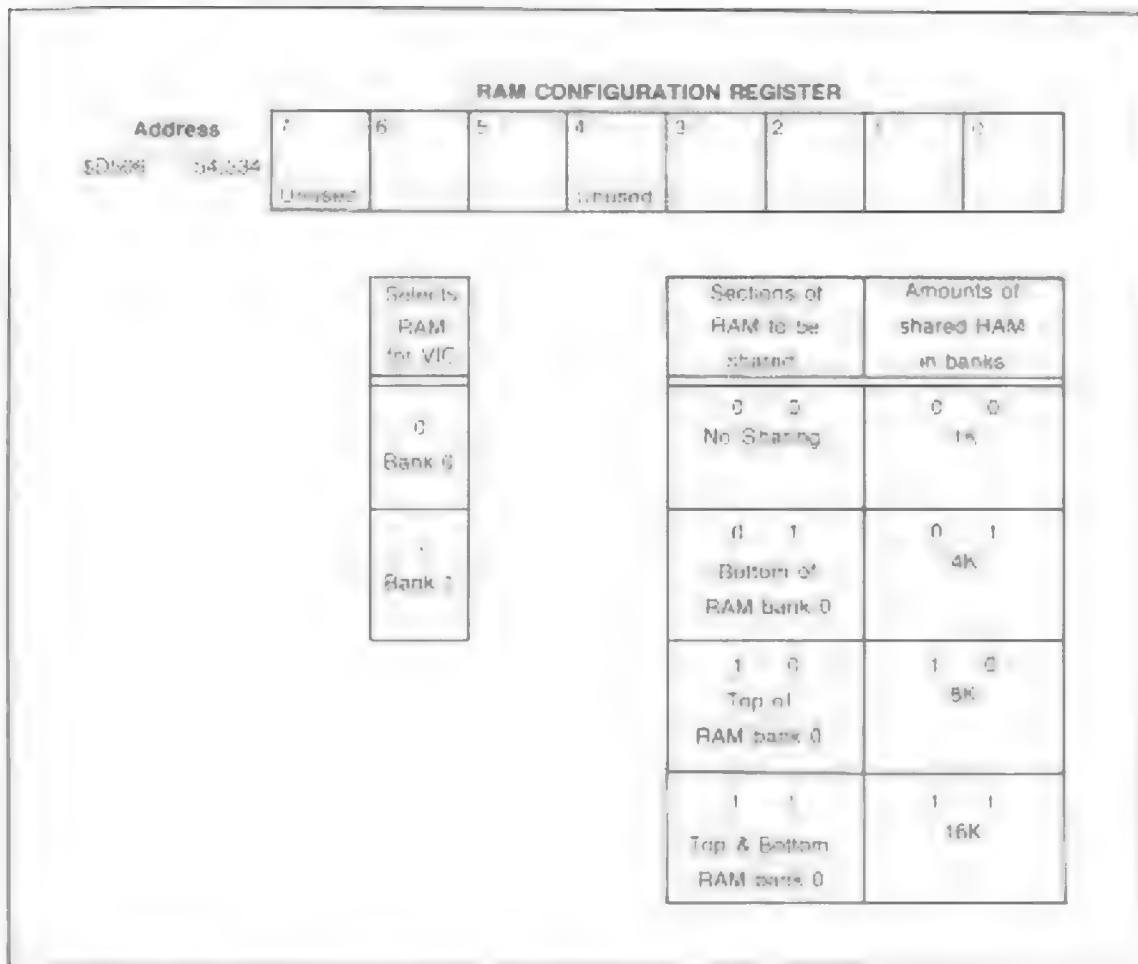


Fig. 12-5 The RCR clears or sets five of its bits to produce the desired RAM sharing.

Bit 6 decides which one of the two RAM banks VIC is assigned to. Bit 7 is unused and also available for future use. If bit 6 is a low then RAM bank 0 will be used by VIC. A high in bit 6 assigns VIC to bank 1.

If you check the RCR contents by typing `M FD506`, then you'll read hex 04. Decimal 04 is binary 00000100. Bit positions are counted from left to right, 7-0. Bit position 7 and 6 are both 0's. Bit 7 is ignored and a 0 in 6 assigns VIC to RAM bank 0. Bits 5 and 4 are also ignored. Bits 3 and 2 are 0 and 1 which places the shared RAM at the bottom of bank 0. Bits 0 and 1 are binary 00 which places the amount of shared RAM to be used at 1K. This

is the arrangement set up at power-up in C128 mode.

THE PAGE POINTERS

Upon system start up in C128 mode, enter the Monitor and type `M FD507 FD50A`. A line of eight hex contents starting at D507 will appear. The F in the hex address shows that bank 15 is the one displayed. The first four hex pairs are the contents of the page pointer registers. The other four hex pairs are the version register (discussed next), and three memory registers in the I/O section of bank 15. This group of MMU registers is located in the I/O section.

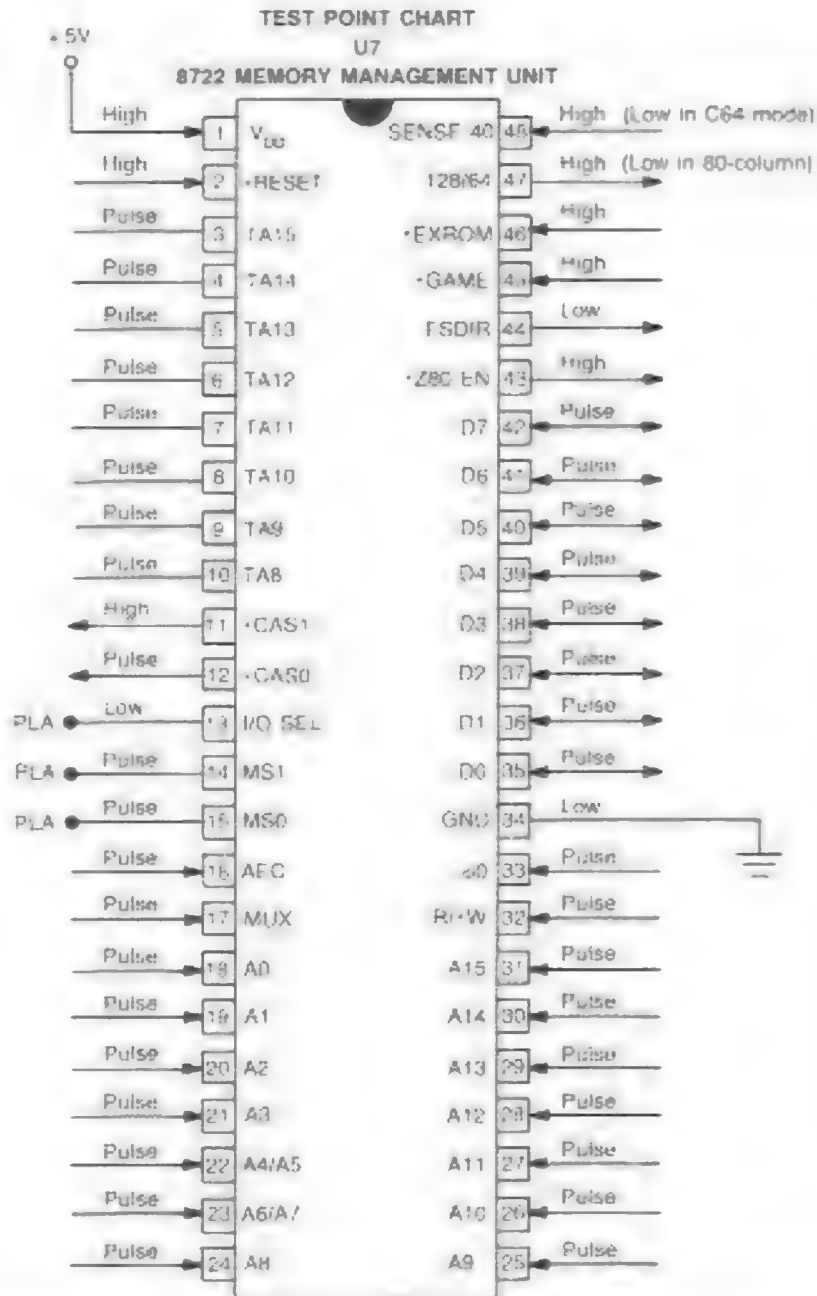


Fig. 15-6 When the computer is down and the MMU registers can't be tested by reading register contents, then the logic probe and the Test Point Chart must be used.

Now a word about pages. 64K of memory, specifically 65536 bytes, is composed of 256 pages of 256 bytes. Each page is composed of 16 groups of 16 bytes. Each page is numbered in hex, from 00 to FF. In the C128 pages 0 and 1 are numbered 00-FF and 100-1FF. Each subsequent 256 byte page is numbered consecutively, in hex, till the last page of the 64K is FF00-FFFF.

The page pointers deal with pages 0 and 1. These are important pages. Page 0 is used by the 8502 because a lot of the instructions it works with are in this page. Page 1 is also vital because it is the C128 stack. That is where the 8502 stores its register contents during interrupts and some Jump instructions.

The page pointers are able to change the page number of 0 and 1 and relocate these pages somewhere else in memory. Registers at D507 and D508, when a relocation of page 0 is performed, will then contain the starting address of the new page number. Registers at D509 and D50A, when a relocation of page 1 is performed, will then contain the starting address of the new page number. Addresses D507 and D509 will hold the low bytes of the new addresses and D508 and D50A will contain the high bytes.

At start-up in the C128, the M command in machine language will show that the four bytes, D507 through D50A, will be 00 F0 01 and F0. These hex values in binary are:

00	00000000
F0	11110000
01	00000001
F0	11110000

The page pointers give the machine language programmer a lot of versatility. For troubleshooting, all that is normally needed is the ability to read the register contents and then compare them to what is supposed to be there at that time. As long as the above values are present, the registers can then be forgotten about. Should an incorrect value be present, that could be a clue that indicates trouble in the chip. That goes for all the MMU registers.

VERSION REGISTER

At address D50B is the version register. The contents of that register in my C128, at system start up in C128 mode, is hex 20. In binary, that converts to 00100000. This register holds an unchanging value that describes the model of the MMU and what and how much RAM is in the machine. It is sort of an electronic name plate, easily changed.

Bits 7-4 contain the code in bits of how much memory is present. The bits 0010 mean that there are two banks of 64K RAM. Bits 3-0 contain the MMU model number or version. The 0000 is the model MMU I have. This register is a signpost for a programmer to know what type of system he is dealing with in case there are some compatibility problems. Computer systems are always being upgraded and inadvertent compatibility problems could be built in. This register could be helpful in those cases.

THE MMU PINOUT

When the C128 has troubles but is still operating, the MMU can be checked out by displaying the register contents and comparing them to the contents that should be there. If the C128 can't display the register contents, then the only recourse to test the MMU is with the logic probe and vom, using Fig. 15-6. The pin states can then be compared with the states shown in the Test Point Chart. If any pins do not match up, then that could be a clue to the trouble.

The Jobs of the MMU

The MMU is connected to the entire address bus, A15-A0. Internally, it takes these incoming bits and translates them into eight translated address lines, TA15-TA8 as shown in Fig. 15-7. The MMU then sends these eight bits and they are used to do other addressing. The TA15-TA8 lines are discussed in detail in Chapter 18.

The MMU is deeply involved in controlling the three processor modes, C128, C64 and Z80. At pin 47 is an output called 128/64. It is a joint output with pin 15 of the PLA. It goes to the character ROM to choose the character set according to the mode.

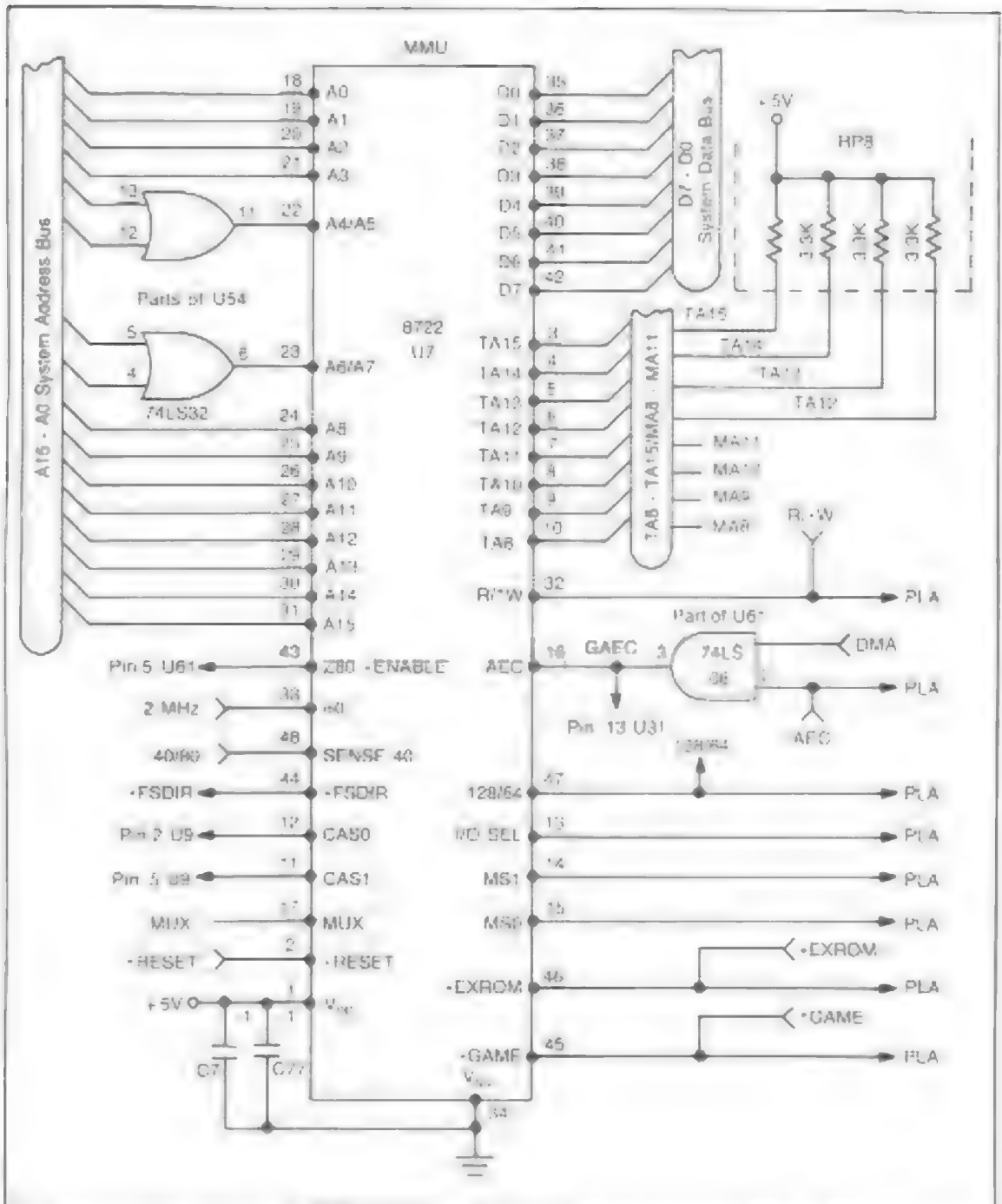


Fig. 15-7 The MMU receives the full set of address bits. It produces the translated address bits TA15-TA8.

C128 or C64. An enable signal at pin 43 makes its way to the Z80 and turns it on and off. It has already been shown how the mode configuration register controls the choice of mode.

The MMU generates select lines for CAS. These outputs are found leaving pins 12 and 11. Other signals such as those to choose the ROM-BANK exit pins 14 and 15, which go direct to pins 17 and 18 of the PLA. The C128 would not be a C128 without the MMU providing all the important control and select measures.

Input Signal

Pins 18-21 are the inputs for address lines A0-A3. The logic probe shows them as pulses. They along with pins 24-31, A8-A15, also all pulsing, are the remaining direct address bits from the processors. Two address bits, A4 and A5, are input at pin 22. At pin 23 are two more: A6 and A7. These pins show pulses too. They are combined simply to reduce the MMU pin count.

These two combined address bits are passed through two OR gates between the bus and the pins. With the use of an OR gate you can combine two signals and save pins and wiring.

AEC, the address enable control from VIC, is input at pin 16. It should probe a pulse. When AEC is high the 8502 is given charge of the shared bus. If AEC is low then VIC takes charge of the bus.

Pin 32 is an input from the 8502. It is the system read/write line. It shows a pulse when okay. When the R/W line is high then the 8502 is ordering a read operation. A low means that the 8502 wants to conduct a write.

At pin 33 there should be another pulse. It is a clock input. It is called Phase 0. The 8502 addressing is triggered on the rising edge and data movement is triggered on the falling edge. More on the generation and use of this signal is given in Chapter 17.

Pin 2 is the •RESET input. When a low enters here, as the machine is turned on or after the reset button has been pressed, the internal MMU registers are all started and initialized. Pin 2 is nor-

mally held high till •RESET, a low, arrives. Pin 2 will probe a high during test standby conditions.

Pin 17 is another pulse. It is MUX, the memory multiplex signal. It is another clock-like input that is coming from VIC.

The other two inputs to the MMU are +5 volts dc from the power supply into pin 1, V_{cc} and ground into pin 34, V_{ss} . Pin 1 will probe a high and pin 34 a low.

Input/Outputs

Because the MMU can be programmed by the processors, it is given normal connections to the data bus. D0-D7 are connected to pins 35-42. They should all probe pulses. The processor uses them almost exclusively to write to the MMU. You can, however, also read from the MMU with these data lines. This is very handy during programming and testing. Lines •GAME and •ENROM are at pins 45 and 46. They should both probe high during test time. They are sensing lines to detect a C64 or C128 cartridge in the expansion bus. When a cartridge is in the expansion port these lines will go low in accordance with the cartridge present.

Pin 48 is the 40/80 column control. It will probe a pulse. When the 40/80 key is pressed the line will tell the MMU what column should be displayed according to whether the key is fixed up or down. As a 40/80 detector, it is an input. There is no output function assigned to the circuit. In the future the output ability could be utilized.

The •FSDIR pin 44 is used to control the direction in which data will flow in the fast serial disk interface. It will probe a low under test conditions.

Output Signals

With all those inputs and I/O signals flowing in the MMU, a group of outputs are generated. Though: a wrong input doesn't usually mean the MMU is defective, a wrong output usually does.

The signals TA15-TA8 exit at pins 3-10. They should all test as pulses. If all the rest of the pins test okay and one or more of these pins are not pulses, then chances are good that you must change

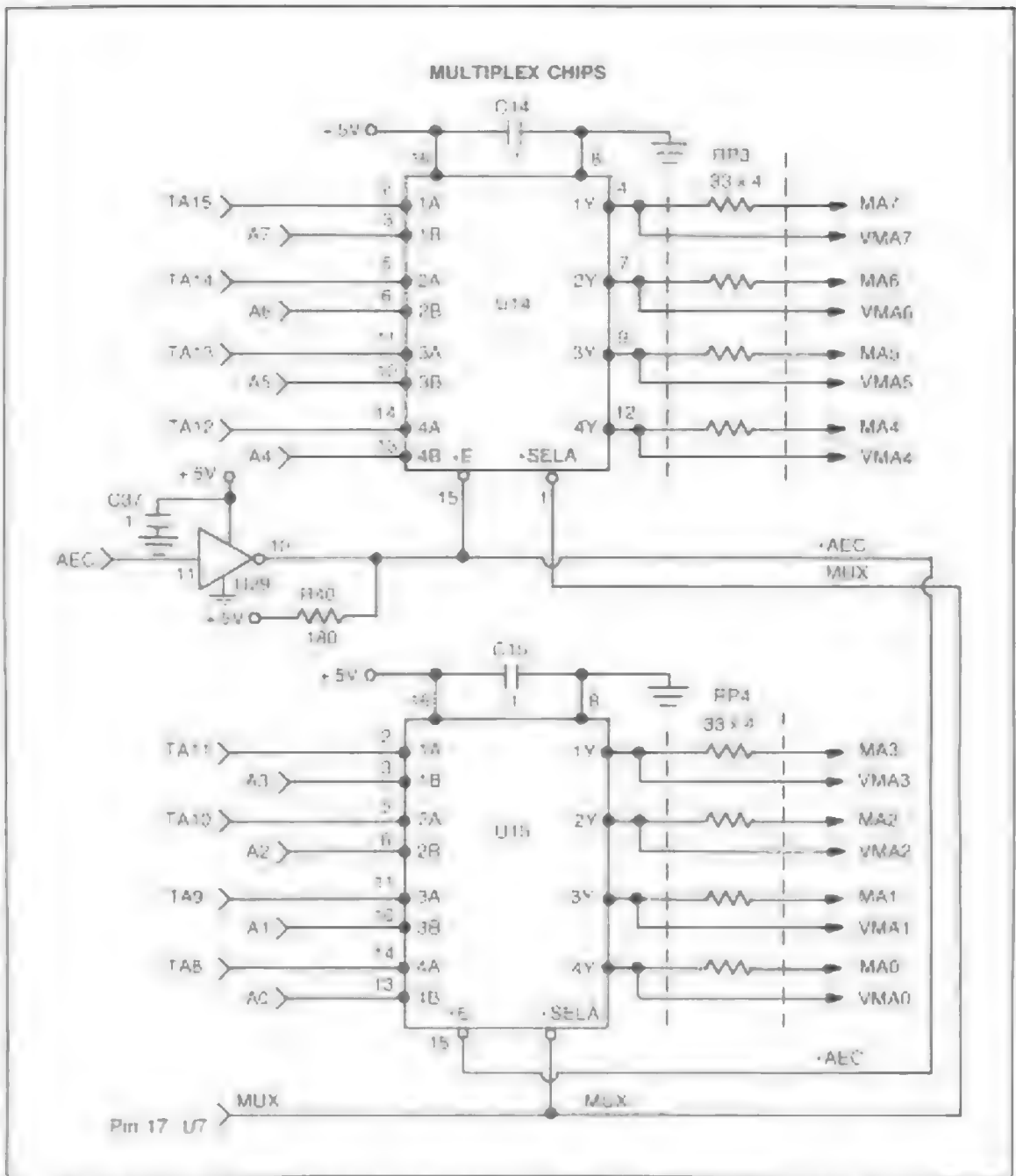


Fig. 15-B. The Translated Address Bus from the MMU connects directly to the Multiplex chips that perform processor addressing of the DRAMs.

the MMU. TA stands for Translated Address. They are sent to the two multiplexers, U14 and 15, in Fig. 15-8, along with other address lines to find locations in the 128K of RAM. They are also sent to the Shared Static address bus that finds locations in the character ROM and the color RAM. During AEC low, when VIC is in charge, these lines are tristated so that they won't interfere with VIC's addressing. More detail on all the address buses is given in Chapter 18.

The column address lines from the MMU are *CAS0 and *CAS1 emerging from pins 12 and 11, respectively. Pin 12 should be a pulse and 11 a high during testing at standby. When CAS0 is a pulse it turns on bank 0, 64K of RAM. At that time CAS1 is held high which turns off bank 1, the other 64K of RAM. When bank 1 is turned on and bank 0 off, the two pins switch states. During standby testing, however, CAS0 is on as a pulse and CAS1 is off as a high. They control which 64K bank of RAM is in use at the moment.

Pin 43 is the Z80 off-on switch. When it is high, the Z80 is off and the 8502 is in charge. If the pin is a low then the Z80 goes on and the 8502 goes off. During standby testing it should be a high but will probe a pulse, putting the 8502 on at that time.

Pins 15 and 14 are MS0 and MS1—two mem-

ory status lines. They should read pulses at the pins. These pins output to the PLA at pins 18 and 17, ROMBANK LO and ROMBANK HI. These two signals after being decoded in the PLA, select desired ROMs in the C128 mode. In the C64 mode these lines do not operate in the PLA. The PLA handles the C64 mode ROM selections without this MMU help.

When both of these lines are low, one of the four system ROMs is selected. If MS1 is high and MS0 low, then the function ROM in U36, when present, is selected. Should MS1 be low and MS0 high, then the external ROM cartridge is selected. When both lines are high, then the underlying RAM, beneath the ROMs with the same addresses, is selected. As mentioned, in C64 mode these lines are meaningless.

Pin 13 also connects directly to the PLA at pin 16. It can be referred to as MS2, and gets the PLA to select I/O addresses in the memory map. It reads low on the probe in C128 mode, and high in C64 mode. As a high, the PLA will not use it. Pin 47 is sometimes called MS3. It also connects directly to the PLA. In C128 mode, it will read high. That is the designation for C128 mode from this 128/64 pin. In C64 mode it goes low. It is the mode control pin from the MMU!

16. All the Memory Maps

With regard to memory maps, three different computers are in the C128. The C128 itself is a special C64 and a Z80 computer. In Siamese triplet style, the three computers are joined together at many places in the machine (see Fig. 16-1). They share components, bus lines and other hardware. They do not share memory maps. Each separate computer addresses its own locations, completely apart from each other. However, even though the mapping is separate, they all use the same field of play.

The same physical residents are represented on all maps. There are 16 DRAM chips, four internal ROMs, one internal functional ROM empty socket, one external cartridge ROM socket, a character ROM, a color RAM chip, two large I/O chips, the MMU, VIC, the Video Controller system and SID.

The 8502 addresses all these chips in various configurations for the C128 and the C64 modes. The Z80 addresses the chips it needs for its Z80 mode. If any of the chips that can be addressed have trouble, then it will cause symptoms in any particular map

in which it is included. This provides a shortcut during servicing. You do not have to test all three modes to find a bad chip. You can concentrate on any one mode that uses suspect chips. When you find the bad chip or chips and replace them you will have fixed all three modes at the same time.

It is probably best to do troubleshooting in the C128 mode as long as the machine is working. Of course, if the computer is completely down and won't come up in any mode, then you can't. When the machine is operating, although faulty, then this approach is valid. The C128 mode provides full coverage of all memory map chips in its 16 bank configurations.

Another important item recommending the C128 mode for servicing is the Machine Language Monitor that only appears in the C128 mode. It is a valuable signal injection and signal inspection device. It has features that permit you to poke signals into memory and then peek to see if the signals have arrived intact. The monitor also is able to read the ROMs. It can also inspect all the registers in any

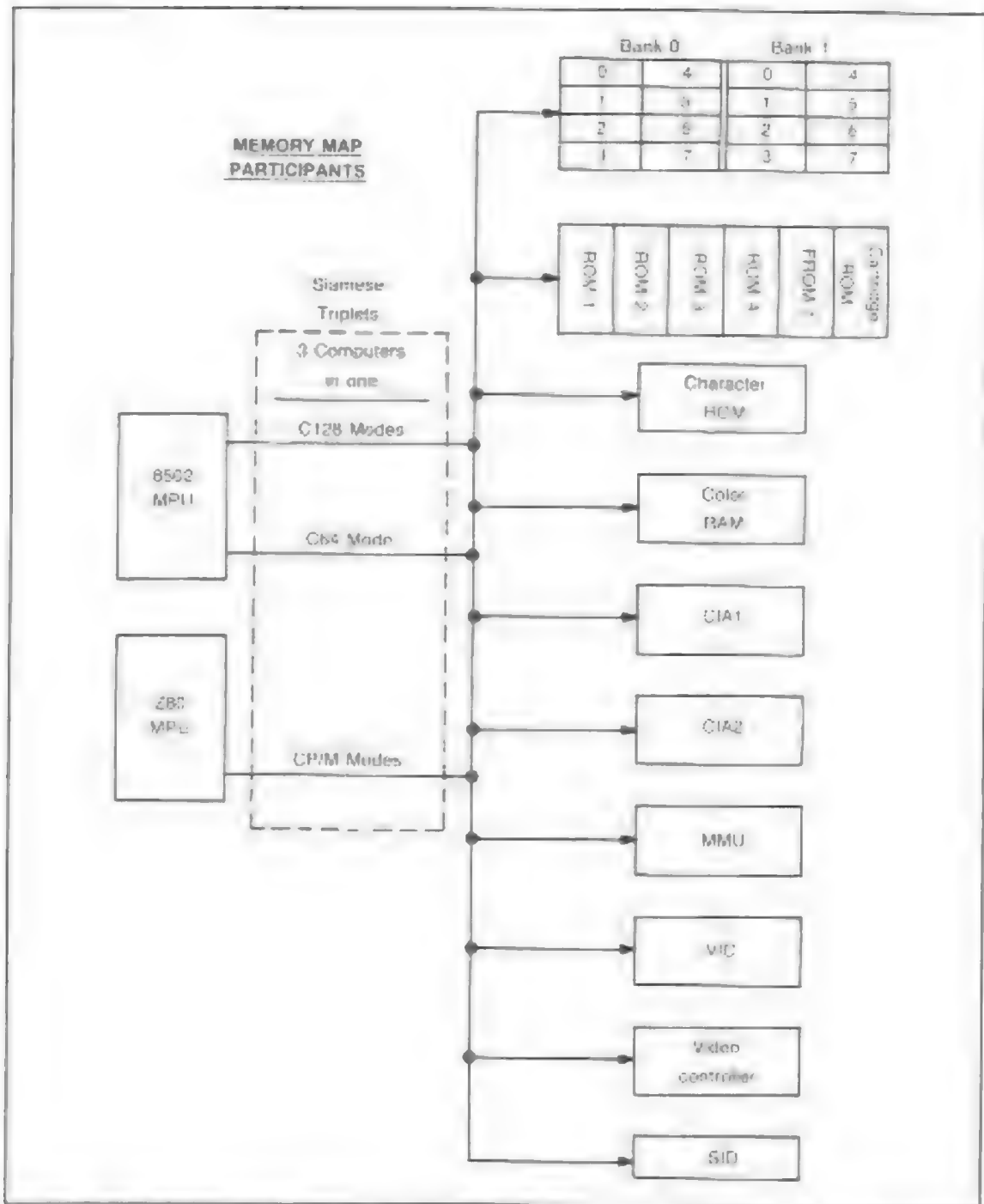


Fig. 16-1 All of the addressable chips are on the right. The two processors and the three modes mix and match them to produce their memory maps.

chip that is on the map. These are powerful servicing abilities. It is almost as if the set designers had purposely installed a signal generator and specially equipped oscilloscope for memory map servicing. Before we get into the signal manipulations with the Monitor in C128 mode, let's look at the banking abilities of the 8502. The 8502's special banking pins are LORAM, HIRAM and CHAREN.

The 8502's I/O Port

The 8502 helps the MMU during banking operations. It has three pins: P0, P1 and P2 that output signals LORAM from pin 30, HIRAM from pin 29 and CHAREN from 28. All three pins are held high with connections to +5 volts through 3.3K resistors, as shown in Fig. 16-2. They go low when they output a signal. The three signals all go to pins 22, 23 and 25 of the PLA. In the PLA chip, they will combine with some other signals to do the following jobs.

LORAM and HIRAM are used to select from the two sections of the color RAM. When LORAM goes low it selects the color that is to be seen while the processor is in control. If HIRAM goes low then the color is selected that is to be seen when VIC is in charge. This ability lets the C128 switch from one full color picture to another without undue picture distortion. Also, the processor can work on a full color picture while showing another picture.

CHAREN has a valuable job. It is used to turn the character ROM off and on in a particular bank that is using VIC. It is turned on when needed and then when it is no longer in use, it gets turned off. The character ROM shares addresses with I/O devices. It is handy to simply bank in the ROM when needed, then get rid of it and let the I/Os use the addresses.

The 8502 pins: P0, P1 and P2 are coming from a special I/O register inside the 8502. This is a byte-sized register, and LORAM, HIRAM and CHAREN are bits 0, 1 and 2. The rest of the bits are used by other functions unrelated to memory maps. Bits 3, 4 and 5 are P3, P4 and P5—cassette controllers. Bit 6, P6, is the CAPSLK. Pin P7 is unused.

Bits 0, 1 and 2 do the memory work with the outputs LORAM, HIRAM and CHAREN. This I/O register has its input connected to the internal data bus, D0-D7, inside the 8502. The internal data bus is connected to the system data bus via pins 38-31. The internal and system buses are directly connected. The locations in RAM could just as well be inside the 8502 if they could fit. However they are actually outside the 8502. I bring up this point because, as shown in Fig. 16-3, the I/O register has its input side connected to the data bus. RAM addresses 0 and 1 are assigned to work with the I/O register. Address 0 is reserved to be the I/O register's data direction register, and 1 is reserved to operate as the input/output port. The I/O register is the peripheral output buffer register. It simply outputs the resultant states of +LORAM, +HIRAM and +CHAREN.

Figure 16-3 shows the RAM addresses 0 and 1 and how they work as the data direction register, DDR, and output register, OR. They are both connected to the eight-bit data bus of the system. As mentioned, though they work as if they were in the 8502, they are physically in RAM at addresses 0 and 1. The restriction is: they must not be used for any other RAM purpose. They belong to the 8502. They are in constant use. There are circuits in the 8502 that control the two locations as if they were in the processor. The rest of RAM is independent of the processor.

As you'll read in Chapter 19 (about CIAs that also have DDRs and ORs), the data direction register controls the direction signal flows to and from the output register. The DDR is programmed bit by bit. The DDR bits are coupled to the respective bit in the OR. That is, bit 0 of the DDR connects to bit 0 of the OR and so on down the line. When you program a bit in the DDR with a 1, the corresponding OR bit becomes a single bit output port. Should you program a DDR bit with a 0, the same bit in the OR becomes a single bit input port. The perspective is from the processor. When a bit is an output, it is an output from the processor to the rest of the computer. An input bit is an input to the processor from the rest of the computer.

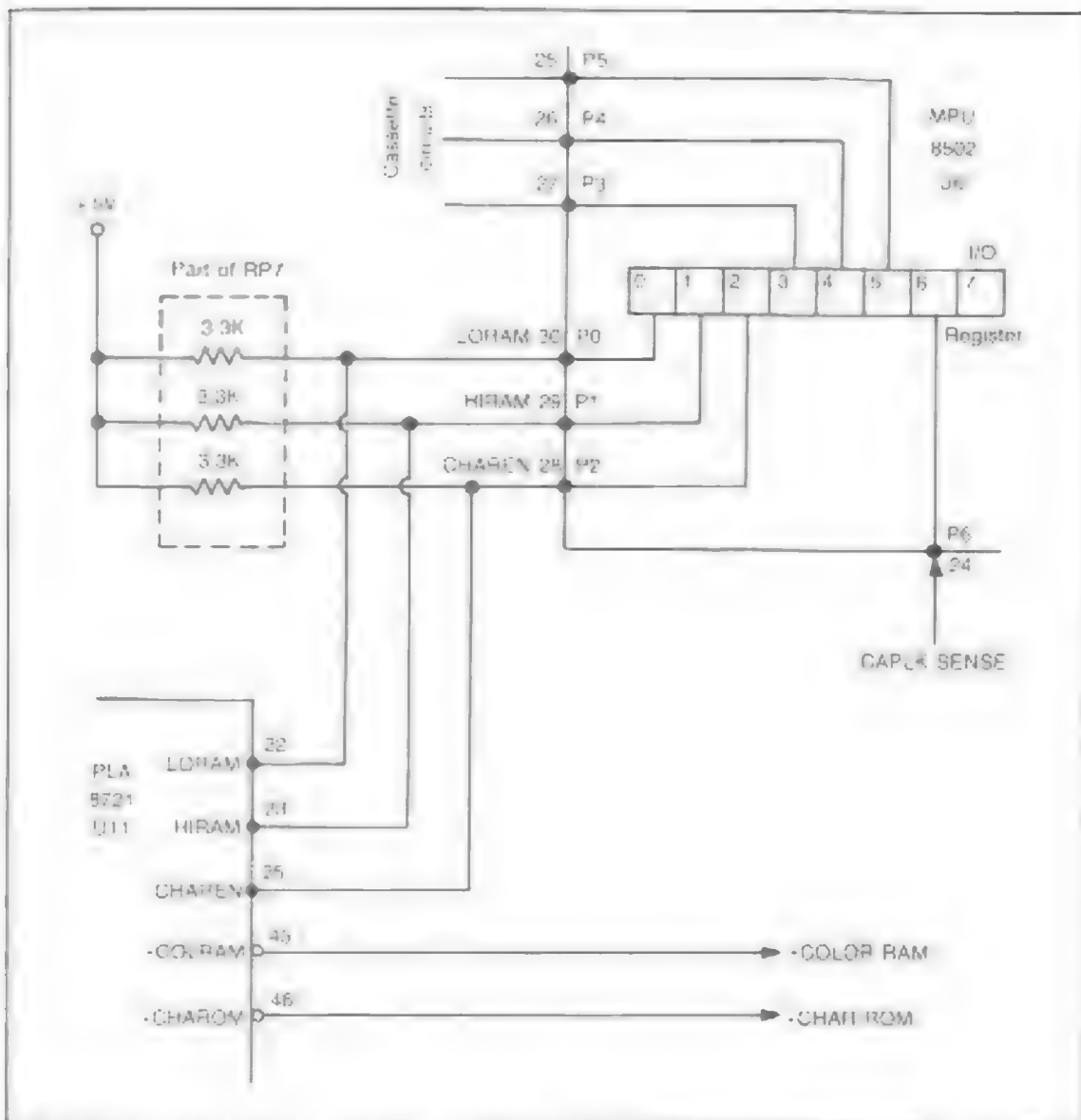


Fig. 16-2 The 8502 has a special I/O register. The three lowest bits output LORAM, HIRAM and CHAREN signals. They proceed to the PLA where they are decoded to perform addressing duties.

A single bit port passes signal through one line. Bits 0, 1 and 2 are all to be outputs. RAM address 0, the DDR, therefore has 1's placed in those three bits. That way, when the processor wants to output •LORAM, •HIRAM and •CHAREN from

processor pins, it sends 0's to bits 0, 1 and 2 of RAM location 1, the OR.

LORAM, HIRAM and CHAREN do an important, but just a small, part of the handling of memory. They do select from sections of color RAM and

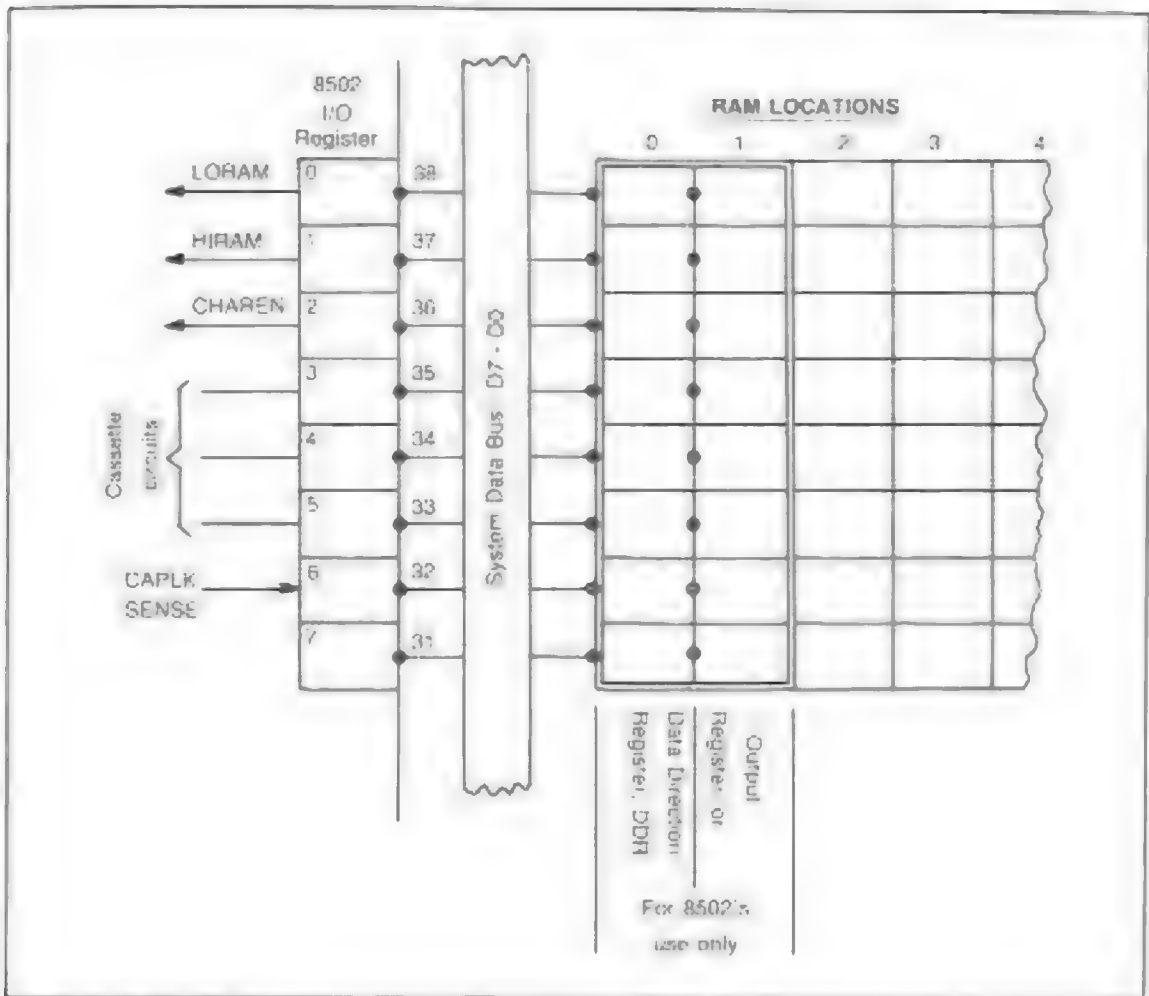


Fig. 16-3 The 8502's I/O register reserves RAM locations 0 and 1. Location 0 becomes its data direction register and 1 acts as the output register.

bank the color RAM in and out of memory space shared with I/O chips. This is only a small area of concern in the vast memory maps.

THE C128 MAPS

The C128 mode makes the most use of all the chips that can be addressed and are thus residents of the memory maps. The 8502 can only address 64K at a time, with a potential of 385K addresses that could be placed into this machine and addressed. The 8502 is able to separate the potential addresses into 16 different banks and address them one at a

time as it switches from bank to bank at an instant's notice.

The layout of all these addresses on all these chips can be thought of as a building with eight stories, as in Fig. 16-4. All eight stories have address numbers ranging from 0 to 65535. The processor can make up a 64K floor by choosing addresses from any of the eight stories and putting them together as one floor.

The bottom four floors are four sections of RAM. The bottom floor is 64K of RAM 0. The second floor is 64K of RAM 1. The third and fourth

C128 MAP LAYOUT

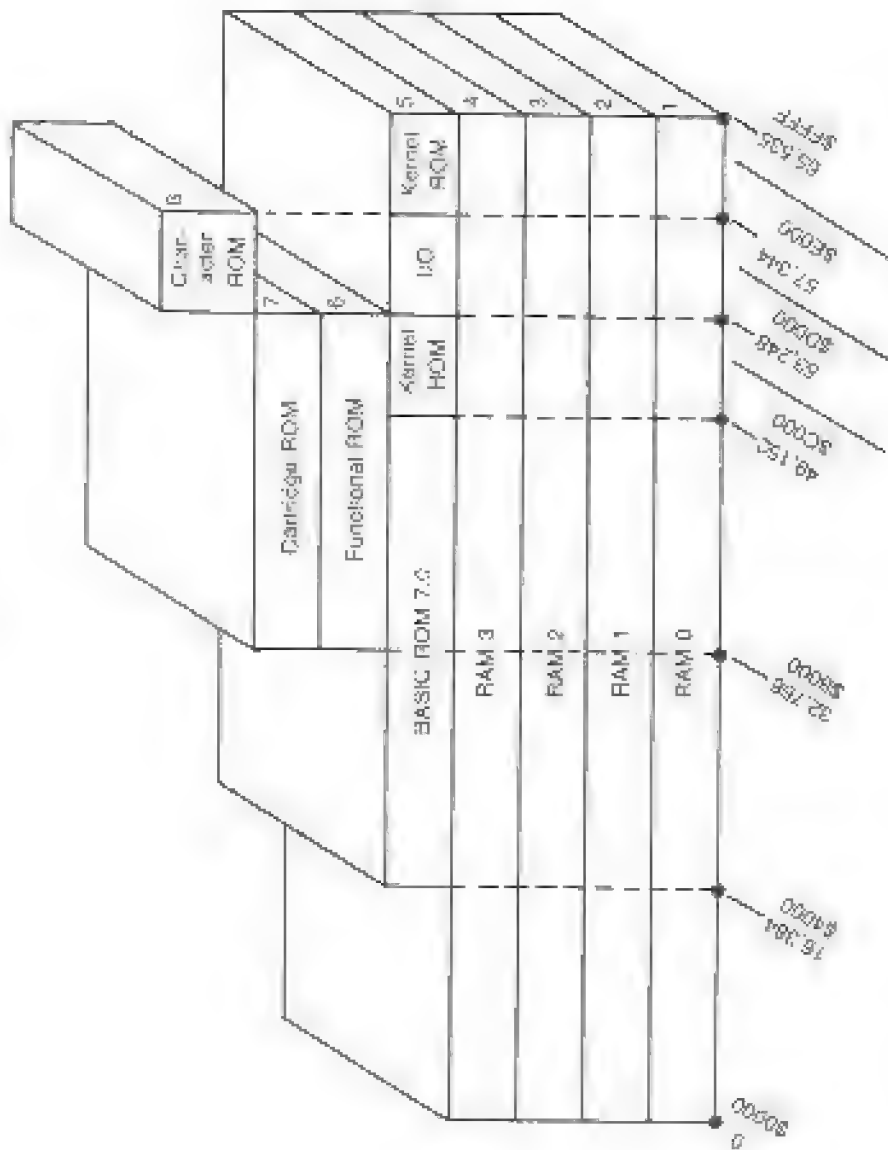


Fig. 16-4 The C128 Memory Map can be thought of as an eight-story building. Each floor has 8K of addresses from 0-65,535 in decimal or \$0000-\$FFFF in hex.

floors are each 64K of RAM 2 and 3 but they do not exist in the present C128. The reason they are included is because the MMU is able to address them in four of its banks. If you do use these banks, the C128 is programmed to give you substitute RAM sections. Instead of RAM 2 you'll be given RAM 0. Instead of RAM 3 you'll get RAM 1.

The fifth floor has the addresses of the four ROM chips that contain the BASIC 7.0, the Kernel, the screen editor and all the I/O. Directly above the I/O is the character ROM addresses in a sparsely populated eighth floor.

The sixth and seventh floors are the addresses of the internal function ROM, U36, and the external cartridge ROM respectively. Figure 16-4 shows the eight-story building and the boundaries of the various address blocks in both decimal and hex. Table 5-2 has more detail.

Because the 8502 can only address 64K at a time, all of the addresses in all eight stories are partitioned out in 16 64K banks of convenience. The 8502 can quickly switch from bank to bank by simply issuing the bank command. The computer comes on in bank 15. Then as the need for a particular chip is required, the bank command is issued and the bank containing the desired chip comes on and the chip can be accessed. The fact that the C128 can switch banks so quick and easy makes it a very powerful machine.

Another feature that increases the versatility of the addressing is the relationship between RAM and ROM. For instance, in bank 15 the 64K residents are RAM 0, the BASIC and Kernel ROMs and the I/O group. Floors 1 and 5 are included in bank 15. This is a two-story bank. The bottom floor is 64K of RAM. Above that is 48K of ROM and I/O. How can the 8502 access this 112K of addresses with 48K having the same addresses?

The machine is built so that if an address is sent out to a ROM location that overlays RAM with the same address, the following happens. If the access is a read, then the ROM is read. Should the access be a write, and the ROM cannot be written to, then the RAM gets the data bits that the 8502 is writing to that address.

It is true that the data bits, stored in RAM because the ROM can't perform in that way, cannot be retrieved in that bank. If those RAM addresses must be read, all that is needed is to switch to another bank where the ROM is not overlaying the RAM. The RAM bits can be read while the new bank is in use.

The I/O chips, also overlaying some RAM with the same addresses, are handled too. The I/O chips have priority over the underlying RAM. The I/O can be both read and written to. The RAM remains dormant beneath the I/O. If the RAM beneath the I/O must be accessed, then here again it is an easy matter to switch to a bank with no addresses over the RAM addresses that need accessing.

As you look over the eight-story building and relate it to the 16 banks, you can see how the banking lets you access all that memory with only 16 address lines.

THE C64 MAPS

In comparison to the C128 eight-story building, the C64 mode in the machine is only a two-story unit with a little penthouse on the roof for the character ROM, as shown in Fig. 16-5. No banks are shown although with the signals, LORAM, HIRAM, GAME and EXROM the 8502 is able to layout at least seven different chip configurations by picking and choosing among the chips.

Programming in the C64 mode is sometimes useful even though the C128 mode is far superior. The inclusion of the C64 mode in the machine was done so that all available C64 software will run on the C128. The C64 mode is quite like the C64 machine itself. There are some differences in the 8502 and the new upgraded VIC. The differences are: the extra P6 pin in the 8502 that handles the CAPS LOCK key; two more registers in VIC that read the extra 24 keys and control the clock rate. These features, however, are used in the C128 mode and not too useful in the C64. A programmer could find them handy in some instances.

The first floor of the C64 building is a stretch of 64K of RAM. Overlaying the RAM is the C64 BASIC ROM addresses, the I/O group and the Ker-

C64 MAP LAYOUT

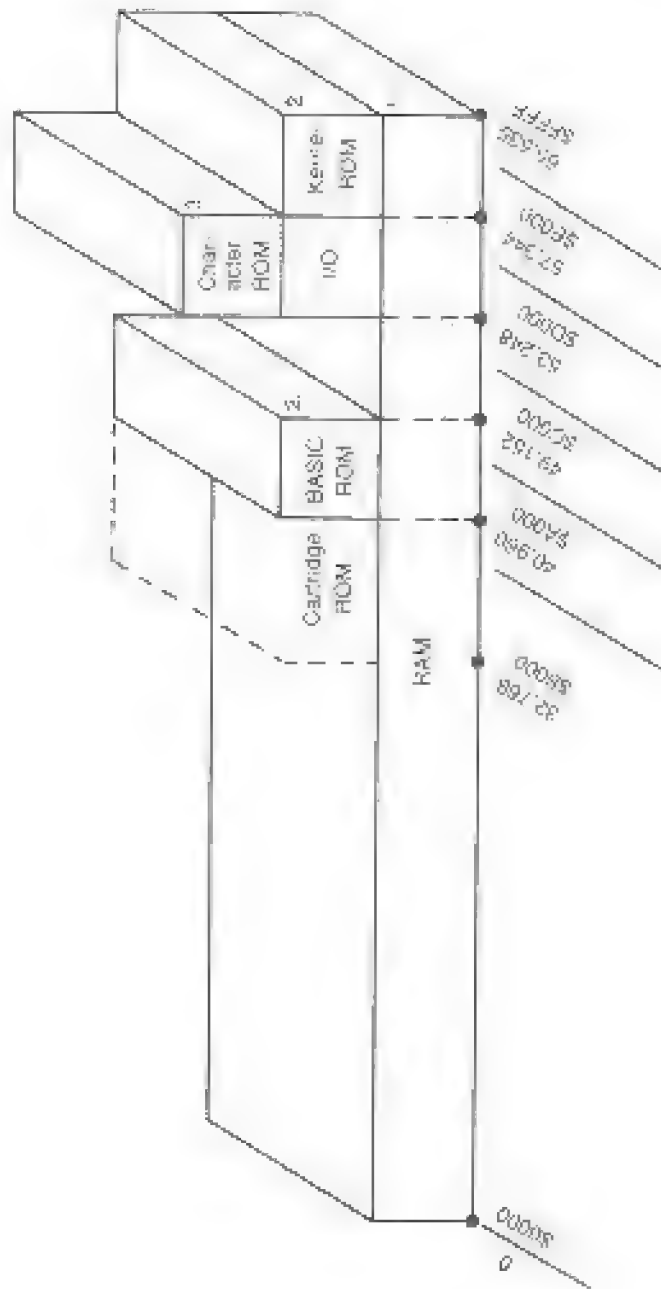


Fig. 16-5. The C64 Memory Map is only a two-story affair with a Character ROM penthouse. The layout can accommodate 64K of the chips in eight different ways.

nel. In the penthouse, above the I/O group, is the character ROM.

A total of 85K of memory chip addresses is in the C64 mode. As mentioned above, the 64K of RAM stretches across the entire first floor. The second floor and the penthouse have the additional 21K. They overlay the RAM from address 32768 to the end at 65535. All of the second floor between 32768 and 65535 are not addresses on chips. A lot of the locations can be empty. The addresses from 32768 to 40959 are reserved for a cartridge ROM. When a C64 ROM is in place, then the area is occupied. The 8K BASIC ROM is at 40960 through 49151.

An unoccupied 4K area is from 49152 to 53247. RAM is beneath that empty area. The RAM can be useful. From 53248 to 57343 is the I/O group, Fig. 16-6. From 53248 to 53294 are 47 registers in VIC. Addresses 54272 through 55295 are 1023 locations assigned to SID. From locations 55296 to 56319, another 1K is where the color RAM is found.

The next chip area is for the CIA1 and CIA2 addresses. The address space allocated for the CIAs is 56320 to 56831. Then above the entire 4K I/O area, with the same addresses but one floor above, is the character ROM.

The final chip on the second floor is the 8K Kernel with addresses from 57344 through 65535.

THE CP/M MAP

The Z80 is very different in makeup from the 8502 in most aspects. The memory map that the Z80 uses to run CP/M programs is also nothing like the 8502-based maps. When the CP/M mode is in use then the C128 and C64 operating systems, for the most part, are turned off. The operating system, which dictates the map layout, enters the machine from the CP/M disk in the external disk drive.

When the CP/M mode is desired, the CP/M disk is placed in the drive and the system turned on. The Z80 as usual gets the machine started and then turns over control to the 8502. The 8502 though recognizes that the disk is in the drive and rather than set up for C128 or C64 mode, it goes into the CP/M mode as a helpmate. It will handle all the disk access or telecommunications that will be needed in CP/M. The first job that the 8502 then does is

load some of the contents of the disk into RAM. Once loaded it then turns control back over to the Z80.

In the CP/M mode, the memory map structure appears as a two-story building as shown in Fig. 16-7. On the bottom floor is the 64K of RAM 0 that is overlaid by some ROM areas. When actual ROM chip addresses are over the RAM, the ROM can be read, but when written to, the bits are stored in RAM, just as the other modes do. In CP/M some ROM is used from the Kernel and a few other spots.

The first 4K in the bottom floor is Kernel ROM that is at locations 53248-57343 in the 8502 memory map. The ROM shows up in CP/M at addresses 0-4095. The ROM performs all the start-up procedures that the CP/M needs. It also contains the program to reset. When the ROM is needed during start up or reset: the Z80 turns off; the 8502 turns on; the routines that are needed are run. Once the machine is started or reset, then the 8502 shuts down and turns control back over to the Z80.

From 4096 to 6144, the keyboard definitions and the 80 column screen storage are addressed. From 6145 to 10239, a part of the operating system is located, called CCP.

A stretch of memory follows from 10240 to 57343. This holds the BIOS and BIOS programs that work with the CCP to operate the CP/M. The rest of the 64K area is full of many vectors and routines to aid in running programs. In this area are the MMU registers at 65280-65284 (hex FF00-FF04) at the same addresses as in the 8502 modes.

On the second floor (another 64K stretch) a long section of RAM is addressed. It is called the Transient Program Area, TPA. It is about 59K long and is used as RAM. After that, the rest of the second floor is taken up by the BDOS and BIOS common areas. The MMU reserved area is on the second floor too.

The letters CCP, BDOS and BIOS were mentioned above. They are the programs that make up most of the CP/M operating system. They are installed in RAM from the CP/M disk. The CCP, Console Command Processor, is the program that gets the programmer into the operating system. It has important control commands. These commands are

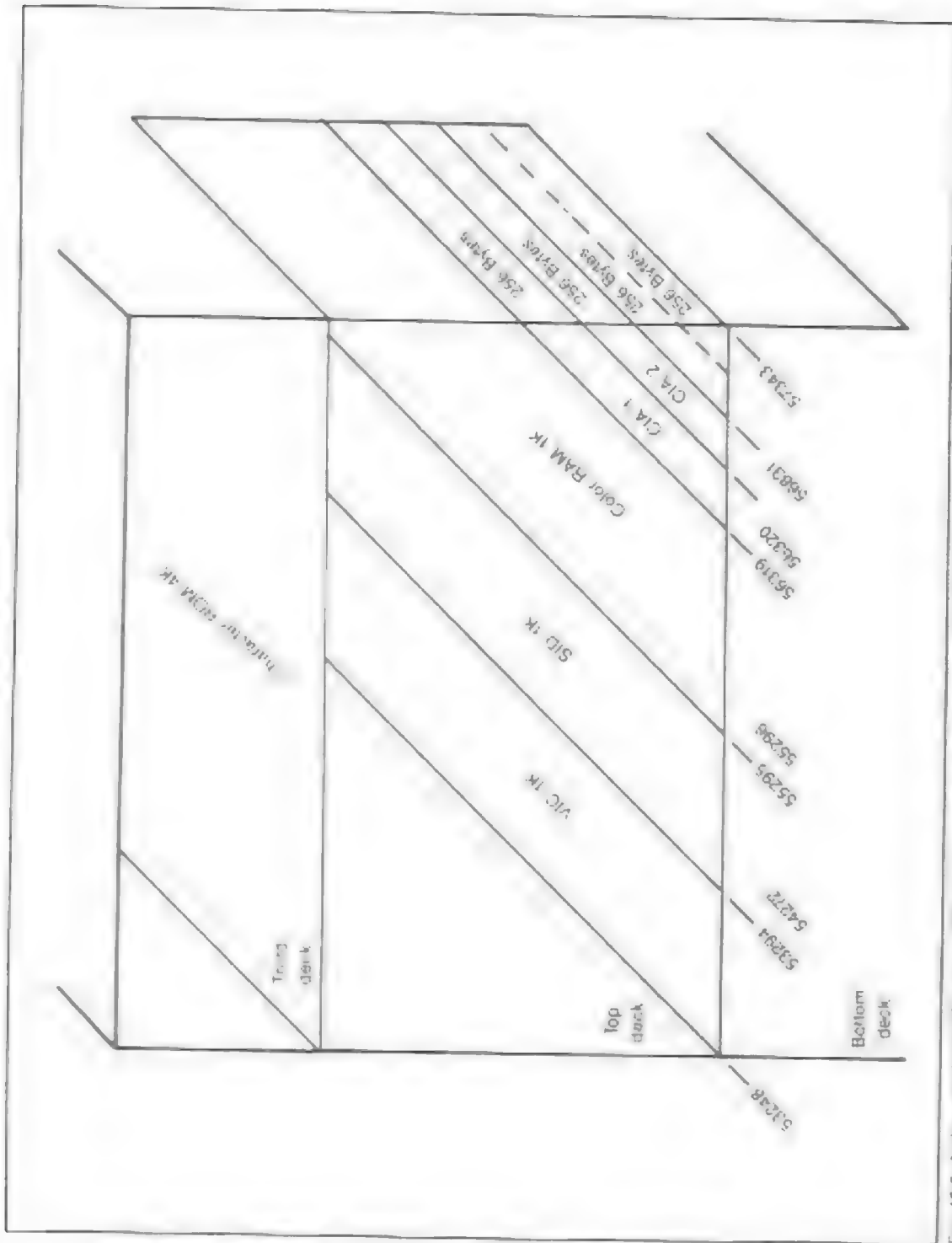


Fig 16-6. A closeup of the 11C section, 53248-57343 shows VIC, SID, the Color RAM, CIA 1, CIA 2 and two unused 256 byte areas.

CP/M MAP LAYOUT

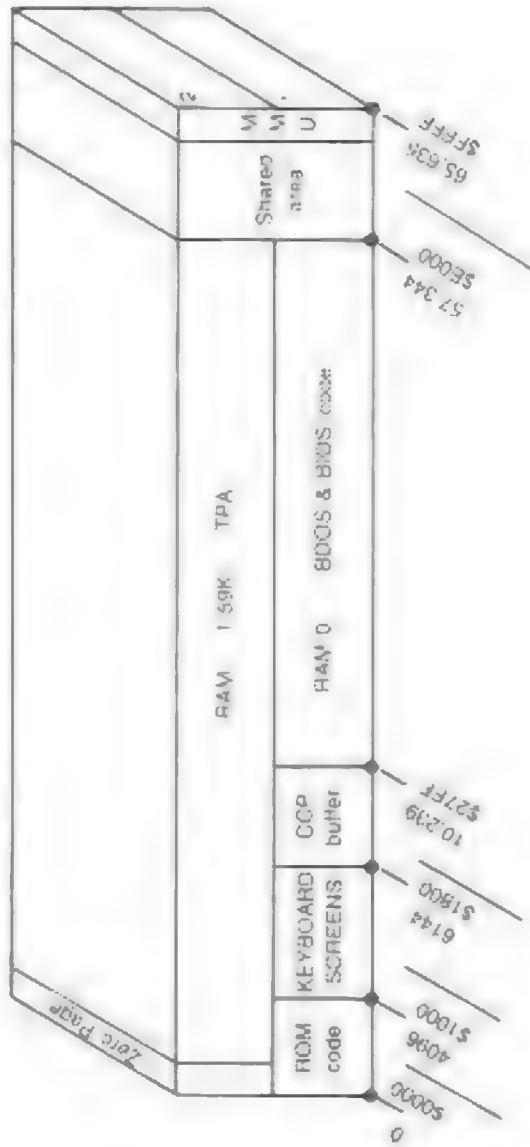


Fig. 16.7 The CP/M memory layout uses RAM 0 to store the operating systems from the CP/M disk and RAM 1 as 59K of usable RAM

TYPE, ERASE, RENAME, USER, DIR and DIRS. The CCP works in the TPA, where all the application programs are performed. The CCP has the Program Loader which loads CP/M programs from disk into the 59K TPA.

The BIOS, BASIC Disk Operating System, is a program that contains functions which the CCP and other programs use to input and output disk and character operations. Also, it is the center of CP/M operations and acts as a file system.

The BIOS, BASIC Input Output System, is also used for I/O interfacing, but handles more complex tasks.

These three sets of programs are important in the running and writing of CP/M programs. If you need more information, the manual that came with your C128 will give you references where more detail can be found. For troubleshooting purposes, these details are not important. The memory map for CP/M, though, could be useful.

PEEK AND POKE

The previous discussions showed how the chips in the memory map areas can be configured in many ways. There are 16 banks available for C128 mode, at least seven maps for C64 mode, and a CP/M map. All these maps use and reuse the same chips. When trouble strikes and a chip is suspected, it is not necessary to use all modes and maps even though the trouble is affecting all of them. If you can find the bad chip and replace it, then all the maps will return.

It would be sensible to use the most convenient and fastest way to do the testing. In the C64 machine, the best way to signal-trace the chips with addresses is with PEEKs and POKEs in BASIC. When you can use them, i.e., if the computer isn't down altogether, they act as a software probe. With PEEK, you can read the contents of the locations on the map. With POKE you can write to any RAM location or any I/O address. POKE is useless with ROMs since they are read-only. They won't accept any more bits.

PEEK is a function that will return an eight-bit binary number which is stored in a register on the map. BASIC will code the bits into decimal and print it on the screen. POKE allows you to write an eight-bit binary number to any RAM location. You enter the decimal code of the binary bits and the command will decode the decimal to binary and install the bits.

When ROM overlays RAM, and you write to a ROM, the bits will be forced into the RAM location beneath the same number ROM location. If you then try to read that location, the ROM bits will be returned not the RAM. All these facts are the way the C64 machine is supposed to work. If it doesn't, then that could be a sign of trouble.

You cannot PEEK or POKE a chip that is not a resident of the memory map. You can only PEEK or POKE an address. All addresses are 16 bits and range from decimal 0 to 65535. BASIC used decimal. To get the contents out of a location and have it printed on the screen, all you have to do is type PRINT PEEK(address in decimal) and then press RETURN. To install a number into a RAM location, you must type POKE(address in decimal) and then press RETURN. These are direct moves and do not require any programming line numbers, although you can write a small BASIC program to do the same thing.

In the C128 machine you can use the exact same PEEK and POKE techniques. They are handy for a quick check of particular locations or in a FOR . . . NEXT routine batch of locations. You can read the values of ROM locations to see if the chip is operating or the location had the correct number. You can write a number to a location or a batch of numbers in a small program to RAM or I/O. Then you could read that location or locations to see if the numbers ever arrived safely. If it did not, that could indicate a defective chip or a bad I/O interface chip.

The BASIC ROM in these cases becomes a software signal injector and tracer. The techniques are indeed useful. In the C128 in C128 mode there is a second device that is even handier. It is the Machine Language Monitor. It does everything the BASIC ROM can do and more efficiently.

MEMORY AND FILL

BASIC uses decimal, but the Monitor uses hexadecimal. The memory map must be addressed in hex. Note that the Monitor only operates in the C128 mode.

To get the Monitor on, either type MONITOR or press the F8 key. You will be greeted by the word MONITOR and the contents of the 8502 programmable registers, as covered earlier. When you enter an address into the monitor, use five digits. The first digit is the bank you want addressed. The 16 banks in hex run from 0-F. After the bank number, use the four digits of the address.

The Monitor has a list of commands that are needed to program in machine language. Two of them, MEMORY and FILL, are somewhat like PEEK and POKE. They are very convenient because you can get a result by simply typing M for memory and F for fill.

With the memory command you can display the contents of any ROM or parts of the ROM. For example, suppose you want to read the contents of the Kernel. It is located at the hex addresses E000 through FFFF. All you have to do is enter:

```
M FE000 FFFFF
```

The entire contents of the Kernel will then scroll down the screen. If you want to see any particular section of the ROM, you can stop the scrolling by pressing the RUN/STOP key as the addresses go by and the section you want appears. If you only want to see one area, for instance:

```
M FE018 FE0C8
```

then one screenful will appear. Try it. Note, on the right-hand side, the row of eight graphic characters representing the contents of the eight hex pairs on the same line. Each hex pair is the ASCII code for its respective graphic. With MEMORY you can look at any section of memory. On each line displayed, there will be the hex contents of eight locations, followed by the eight graphics.

If you want to read the character ROM, all you need to type is:

```
M ED000 EDFFF
```

and the contents of the ROM will scroll past. Bank 14 was contacted this time with the first address digit E. E is hex for decimal 14. The character ROM only appears in bank 14. If you had typed:

```
M FD000 FDFFF
```

then the contents of all the registers in the I/O block would have scrolled past. That would have included some RAM register contents that are beneath the I/O block where there aren't any I/O registers. While MEMORY acts like a super PEEK command, FILL is a super POKE. For instance, a good way to test RAM chips is to exercise them. An exercise could consist of first looking at a section of RAM. In bank 0 there are 64K of RAM and little else. A short section of RAM picked at random is D508-D567. If the command:

```
M 00508 00567
```

is given, then the screen will display the contents of 12 rows of eight registers. Following the register contents are the eight graphics on each line. The unused RAM registers should display either 00 or FF. They are, of course, hex for binary 00000000 and 11111111. The graphic for eight 0's in the C128 is a period and the graphic for eight 1's is the Greek pi.

One little exercise that the Monitor allows is to change a register's contents at will. Using the cursor controls at the top of the keyboard, place the cursor somewhere in the middle of the display on a register. Then type 54 and press RETURN. The 54 will replace that chosen register and the capital letter T will appear in the corresponding graphic spot.

The number 54 is the hex ASCII code for T. What you have done is poke the 54 into the RAM register. The VIC then came along and converted

the 54 to the display as a T. If you then put hex 45, 53 and 54 into the next three registers, then the word TEST will appear in the graphic section. If you can do that, then the eight RAM chips in bank 0 have performed okay. Remember, there is one bit in each register on eight chips. They all get tested in this way at the same time. Should the test not work, that is a good clue to trouble. Chances are there is a bad RAM chip.

The above test is without using FILL. With FILL you can poke entire RAM sections with bits. The following is a little RAM stress test. We can use the same display. Type the command:

```
F 0D508 0D567 FF
```

and note the FF on the end. This is 11111111 that will be installed in every register between D508 and D567 in bank 0.

When you press RETURN there is no action in the display as the registers are filled. Then when you view them with:

```
M 0D508 0D567
```

you'll see the display fill with all FFs and the graphics

with all pi's. You have shown that this RAM area is able to be filled with highs. The next step in the exercise is to fill them with lows:

```
F 0D508 0D567 00
```

Again, when you press RETURN, only the cursor moves a line. Then the command:

```
M 0D508 0D567
```

is entered. The display moves and the registers addressed in bank 0 are filled with lows. The graphics change to all periods. If all of the highs can be installed and then all of the lows can be made to replace the highs, then the RAM section is operating and odds are these eight chips are trouble-free.

To exercise the other eight RAM chips, simply address bank 1 and perform the same type of test. If you have the time you could run a test of the entire RAM but it would take hours. Sample tests like the above are valid most of the time. If the RAM cannot perform the easy exercise, you could have a bad RAM chip or chips. On some rare occasion some weird trouble would not be found, but that does not happen very often.

17. The Clocks

The master clock in the C128 is based around the clock chip, U28, the 8701 16-pin DIP. The chip is the drummer for the marching of bits that takes place in the computer. As the chip beats out its frequencies, the bits march throughout the machine.

Figure 17-1 shows the schematic connections and immediate circuitry for the clock chip. The chip has a crystal connected across pins 13 and 14. When voltage is impressed on the crystal it starts oscillating at one of two frequencies. In America the clock runs at 14.31818 MHz. This frequency can be divided into all the frequencies needed to conform with American NTSC standards. The other master frequency the oscillator can run at is 17.73447 MHz. This frequency can be divided into the required European PAL standards. There is a jumper coming off of pin 7, ϕ PAL, of the clock chip. When the jumper is open the crystal runs as 14.31818 MHz. If the machine is to be used in Europe then the jumper is connected and the frequency modified for PAL.

Connected to the crystal at pin 13 to ground is a tiny variable capacitance, C20, a 4-40 pF. It can be adjusted and will change the oscillator frequency a small amount. The frequency can be checked with a frequency counter. Connect the input of the counter to pin 8, ϕ COLOR, of the clock chip. Adjust C20 for the exact frequency of 14.31818 MHz in the U.S. In Europe tune for 17.73447 MHz.

The chip outputs are at pins 8 and 6. Pin 8 sends a signal, called ϕ COLOR, to pin 29 of VIC. Pin 6 sends a signal called ϕ IN to pin 30 of VIC. The most striking difference between the frequency generated by the crystal circuit and the subsequent clock output is: the crystal produces a sine wave and the chip changes the sine wave and outputs to a square wave.

SINE WAVE TO SQUARE WAVE

The input circuit to the clock chip is a sine wave produced with the help of the crystal. The crystal is only part of the oscillator circuit. The rest is inside the chip. The sine wave is an analog signal. Dig-

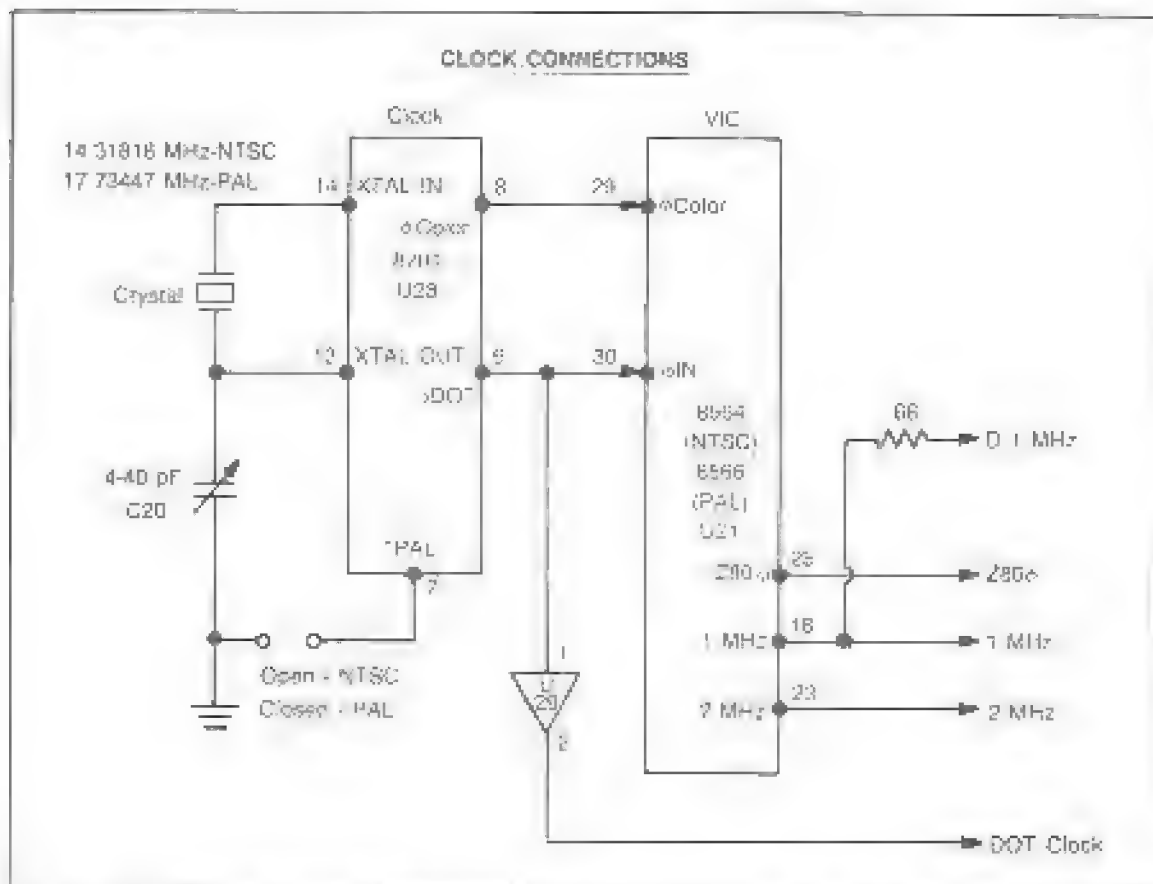


Fig. 17-1. VIC is the engineer of the two clocks that run most of the computer. The only other clock is part of the 68-column circuit covered in Chapter 21.

ital computers, cannot use sine waves. They can only operate with square waves. A sine wave is continuous with an infinite number of smooth changes in one cycle, as shown in Fig. 17-2. A square wave, on the other hand, consists of one low, a change upwards, one high and then a second change downwards. The digital circuit is built to handle square waves with highs and lows only.

The clock circuit, besides being a crystal oscillator circuit is also an analog to digital converter, changing the sine waves to square wave pulses. The frequencies remain the same, only the waveshape is changed.

One cycle of the perfect square wave is thought of as having the following timing. The wave could

start with a duration of half the cycle at the low voltage. At the end of the low duration, the voltage goes straight up to the high voltage with no time elapsing. Then for the duration of the rest of the cycle, the voltage remains high. As the cycle ends, the voltage falls to a low, again instantly, in no time at all.

While the above is the description of a theoretical square wave, it is not possible for the voltage to change from low to high or high to low in zero time. There must be some time used to make the transitions, no matter how small. The time could be measured in billionths of a second—but it is time.

The actual waveshape slopes to the right and upwards as it goes from low to high. It slopes to the left and downwards as it goes from high to low.

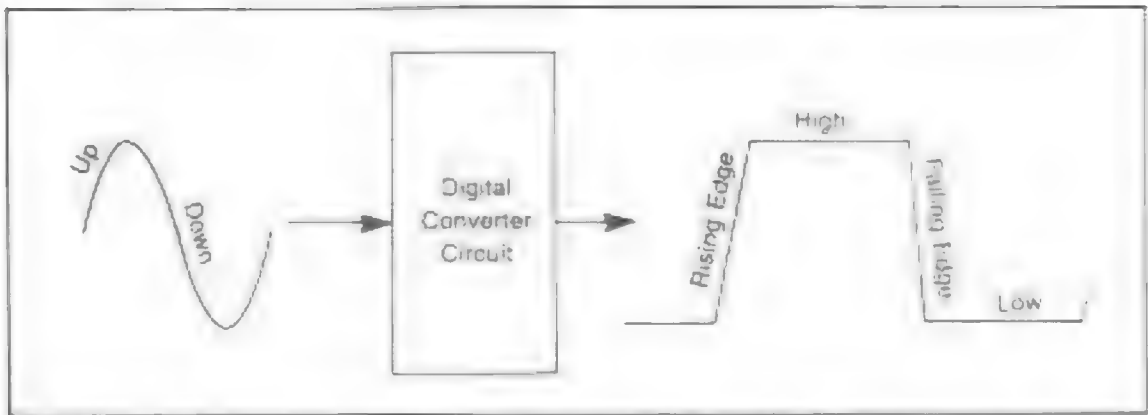


Fig. 17-2 The crystal produces an analog sine wave. The clock chip converts the sine wave to square waves that the digital circuits need to operate with. One square wave consists of a rising edge, a high, a falling edge and a low.

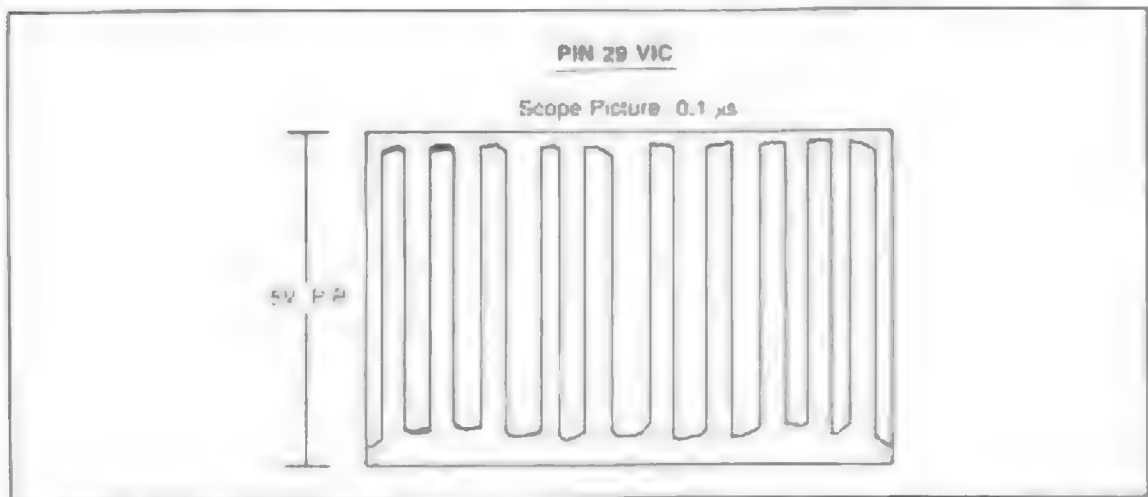


Fig. 17-3 An expensive scope set at 0.1 microsecond sweep will show about ten square waves of the color clock.

The low to high is called a rising edge, and the high to low transition is called a falling edge. The edges are vital in digital circuits to trigger a lot of the activity.

The sine waves are generated at the crystal oscillator circuit, converted to square waves inside the clock chip and output from two clock pins to two VIC pins. These input square waves have names. At VIC pin 29 the signal entering is called ϕ Color, the color clock. At VIC pin 30 the signal is called ϕ IN, the dot clock.

If you are equipped with a good scope you can view the color clock and the dot clock. At a 0.1

microsecond scope rate, at pin 29, Fig. 17-3, you'll see about ten square wave pulses. At pin 30, Fig. 17-4, there are about six pulses. The pulses will have a peak-to-peak voltage in the neighborhood of 4-5 volts. The scope is really not that essential. The logic probe will read PULSE at both of the test points. This is a good test point to determine if the clock is running. No pulse, of course, indicates that the clock is off.

Inside VIC

The square wave enters VIC and proceeds directly to the VIC clock generating circuits. The

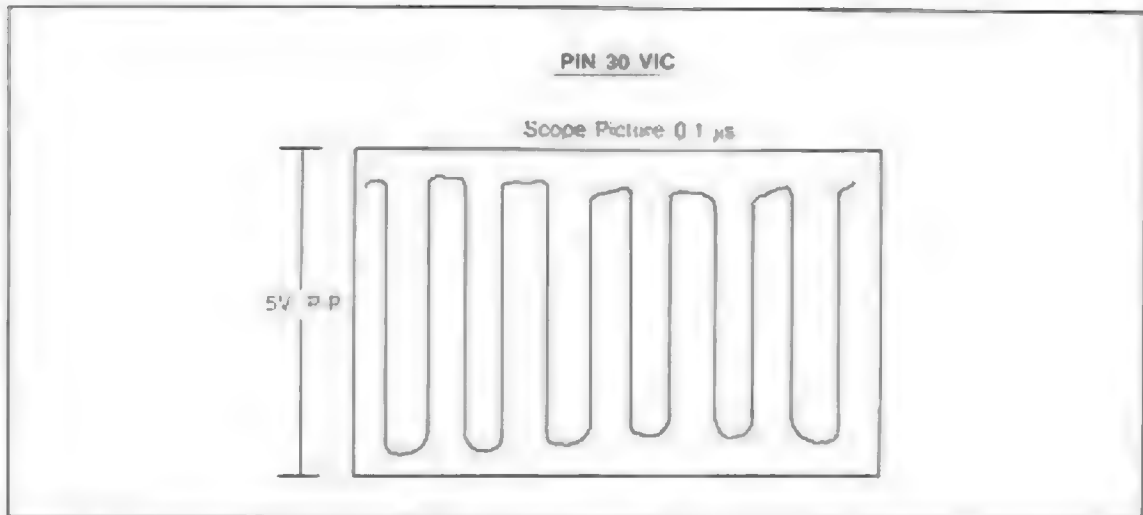


Fig. 17-4. The scope will show about six square waves of the dot clock.

two square waves also go to other circuits to produce the sync/luminance and chroma display outputs, but this is covered in Chapter 20. The system clock signals are there to drive the internal digital circuits.

The first clock generated is the 1 MHz clock. It is made by dividing the incoming clock frequency. This makes the clock approximately 1 MHz. This is the system bus clock and exits VIC at pin 18. The 1 MHz first of all is connected to SID, the CIA2 and the Expansion Port. Then a tap is connected to the 1 MHz clock line, a 68 ohm resistor is placed in series and the resultant signal is called DIMHz. The DIMHz is sent to AND gates to work with the AEC signals. AEC and the 1 MHz control and clock the bus line actions. It also goes to CIA1 and the U12 latch in the Z80 data bus interface.

Out of VIC pin 23 goes the 2 MHz clock. The 2 MHz clock is produced by also dividing the frequencies. The 2 MHz clock is changeable. It is able to run at either 1 MHz or 2 MHz. In fact it could run at 1 MHz a lot of the time. It runs at 2 MHz only under the following condition.

Bit position 0 of a register in VIC at address hex D030, decimal 53296, sets up the 2 MHz clock frequency. If the bit has a high, then pin 23 will output 2 MHz. Should the bit hold a low, then pin 23 outputs 1 MHz. There will be more about this register in Chapter 20.

The so-called 2 MHz clock has a number of destinations. First of all it clocks the 8563 video controller chip. It also proceeds directly to the MMU and the 8502 processor. With its versatile ability to run at either 1 MHz or 2 MHz it has a great deal of control over the computer.

The third VIC generated clock is called Z80 ϕ . It is a special 4 MHz clock that is designed to operate the Z80 processor. It comes out of VIC pin 25. More about this clock will be given later in this chapter.

8502 TIMING CONTROL

The so called 2 MHz clock from VIC goes to pin 1 of the 8502 processor. It is called ϕ IN. As mentioned, the 2 MHz clock runs at either 1 or 2 MHz according to the state of the 2 MHz bit in VIC. Let's examine what happens in the 8502 when it is driven by the 1 MHz mode.

The 1 MHz pulse enters pin 1 and goes directly to the Timing Control circuits in the 8502. Fig. 17-5. It is the only input to the circuits. The circuits take the incoming pulses and generate three outputs. The timing circuits are based around a phase detector circuit. In this circuit, the incoming ϕ IN is split into two pulses, both still running at 1 MHz but made out of phase with each other by 90 degrees. They are called ϕ 1 and ϕ 2. These two ϕ signals should

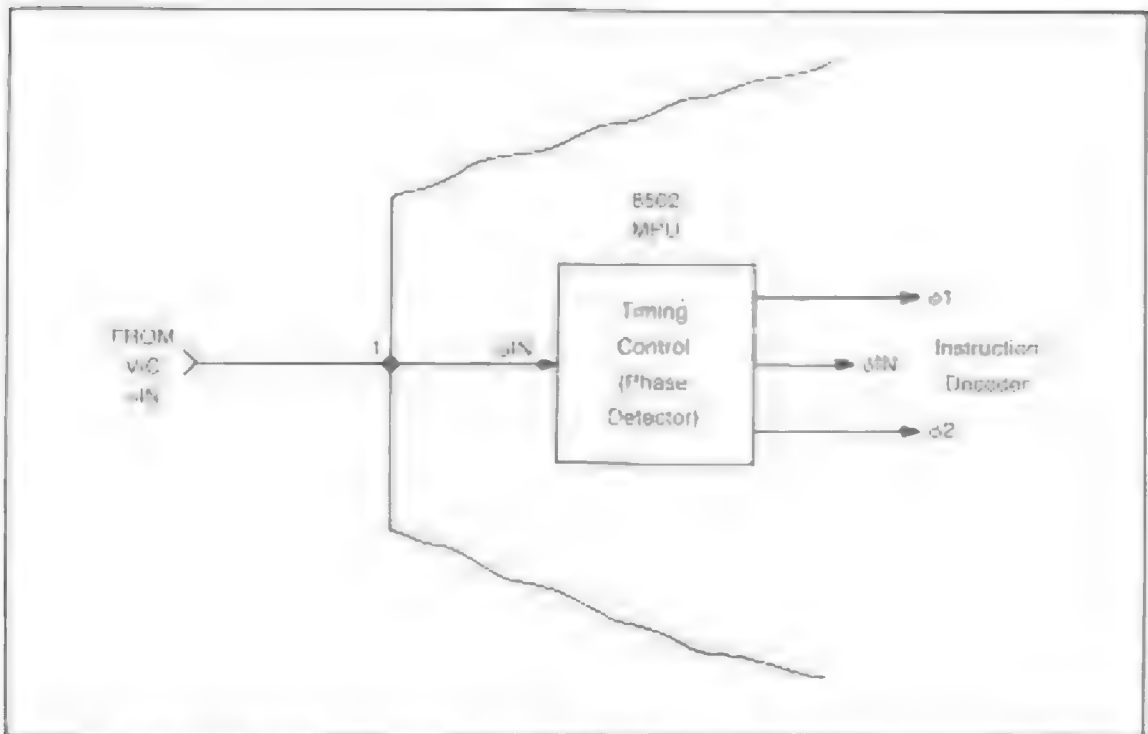


Fig. 17-5 A one MHz goes to the timing control circuit in the 8502 processor.

not be confused with any other ϕ signals in the computer. These are internal to the 8502.

$\phi 1$ and $\phi 2$ are then input to the Instruction Register and enter the decoder section of the register circuits. The decoder matches the way the address and data bus pulses are running. $\phi 1$ is matched to the address bus pulses and $\phi 2$ with the data bus pulses. $\phi 1$ is going to be the drummer for the address bus and $\phi 2$ the drummer for the data bus. The bits in these buses will march to the beat of their special drummer.

$\phi 1$ and $\phi 2$ thus become the working frequencies of the 8502. Figure 17-6 shows the timing of the signals. The two signals are quite alike except their timings are out of phase. This makes their highs and lows staggered. The sketch shows that $\phi 1$ has a high while $\phi 2$ has a low. Then when $\phi 2$ goes high $\phi 1$ goes low. Note the slanted rising edges and falling edges that represent the short time that elapses between the highs and lows.

Figure 17-7 illustrates one complete cycle at 1 MHz. That means the cycle takes place in one millionth of a second. Because 1000 billionths of a second are in one millionth of a second, and billionths are called nanoseconds, the cycle is taking 1000 ns. The ns is the conventional measurement of the clock cycles. The high and low represent the voltage involved.

The important part of the cycle, to the computer, is the duration of the high in the cycle and the quick voltage change of the edge. The low really doesn't do much but mark time while the highs and edges are producing the computing. The high in $\phi 1$ lasts for a duration of 430 ns. The low in $\phi 1$ takes place in 470 ns. The slanty rise and fall edges take place in 25 ns.

What are the highs and edges doing? Let's take a trip along with one cycle of $\phi 1$ and $\phi 2$. The cycle begins at the rising edge of $\phi 1$. Remember that the instruction decoder, where the signals are operat-

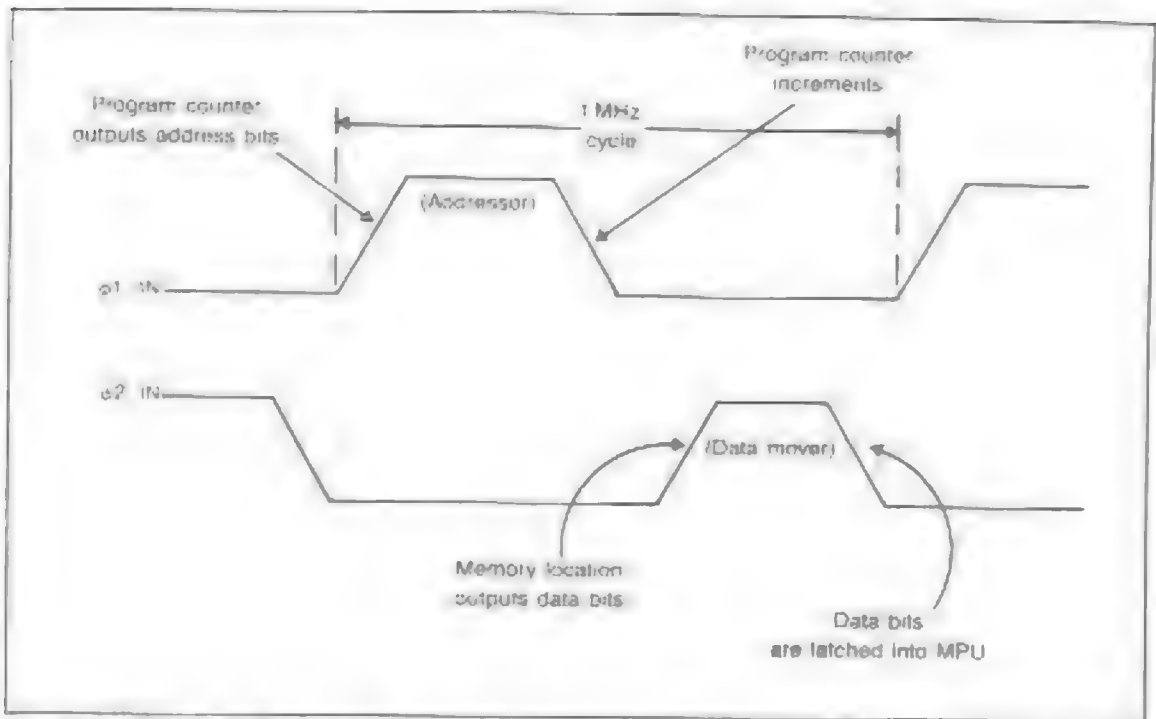


Fig. 17.6 The timing circuits output two 1-MHz clocks called $\phi 1$ and $\phi 2$. Clock $\phi 1$ drives the program counter and $\phi 2$ operates the data bus.

ing, is connected inside the 8502 to all the registers and internal bus lines of the processor. As the cycle starts the rising edge of $\phi 1$ is changing from a low to a high in 25 ns. The fast changing voltage has a trigger effect.

As mentioned, $\phi 1$ is concerned with addressing. Therefore the rising edge triggers the program counter. It forces the PC to place the current address in the 16-bit register to place the bits onto the address bus. $\phi 2$, meanwhile, is low and not doing much. Note the falling edge of the previous $\phi 2$ cycle has hit bottom right before the rising edge of $\phi 1$ took off.

Once $\phi 1$ puts the address bits onto the address bus the high duration of $\phi 1$ takes place. It lasts for 430 ns. This is plenty of time for the bits to travel the length of the address bus and open up a location on the memory map.

During the 25 ns that the $\phi 1$ falling edge takes, the PC is triggered by the shock of the fast voltage change from high to low. This trigger makes the PC

increment itself by one. This sets up the next sequential address. That way the PC is up and waiting for the next cycle.

As $\phi 1$ begins its low, the low in $\phi 2$ comes to an end. The rising edge of $\phi 2$ begins. The 25 ns rise triggers the 8502 to place the byte of data it is interested in onto the data bus. If the operation is a read, the data in the addressed location is placed on the bus. Should the operation be a write, the data the 8502 had been working on is placed on the data bus. Either way the $\phi 2$ rising edge places data on the bus.

Next the duration of $\phi 2$ takes place, for 470 ns. During that time the data is well able to travel from the memory to the 8502 or from the data bus buffer in the 8502 to memory. As the $\phi 2$ duration ends, the falling edge occurs. It causes the data from memory to be latched into the Instruction Register of the 8502 during a read, or the data from the 8502 to be latched into memory during a write.

The 8502 knows to ship data to memory when

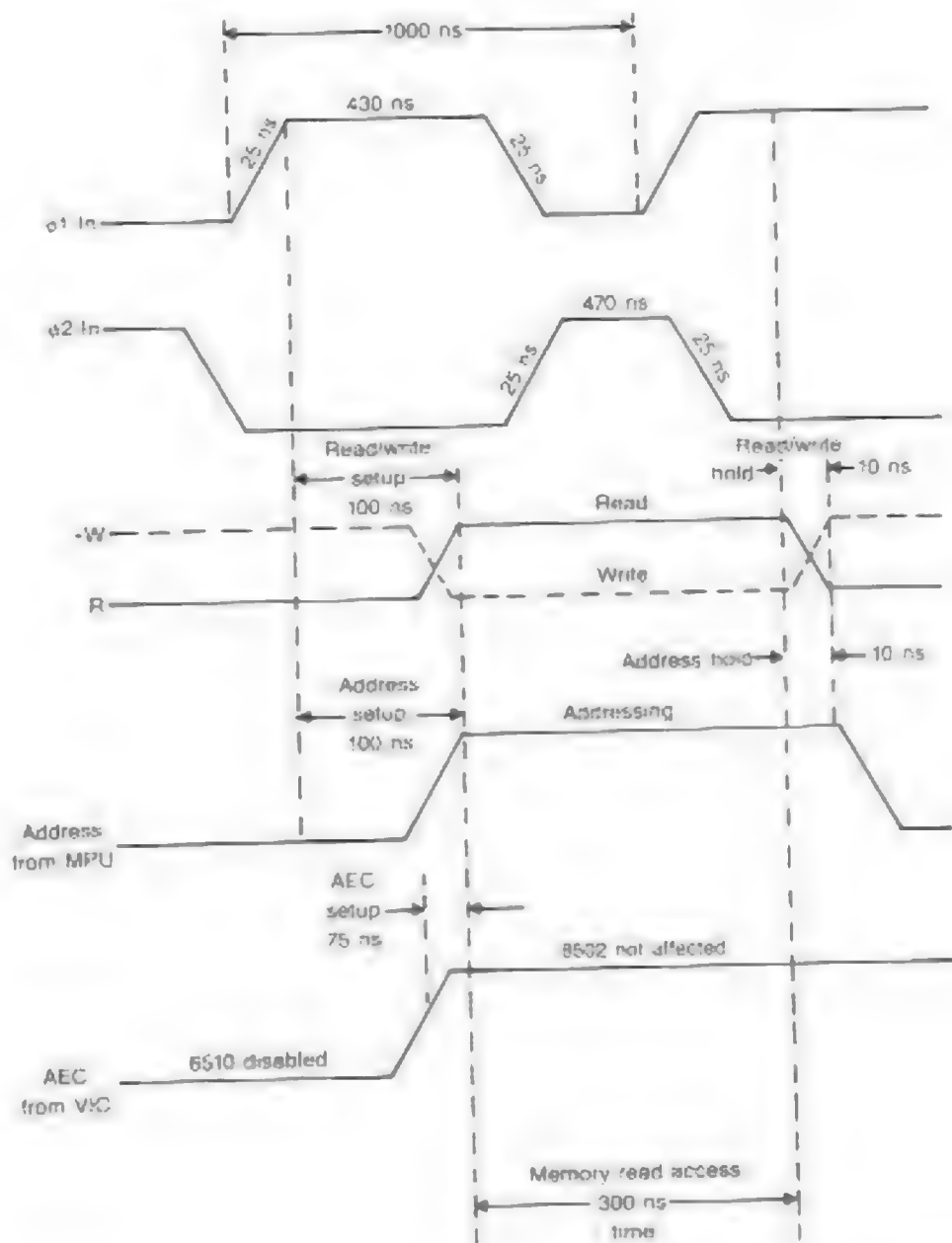


Fig. 17-7 The $\phi 1$ clock is the signal that gets the addressing done. Note it is high when the Read goes high or the \bar{W} goes low. Clock $\phi 1$ is also high when the address bits from the MPU enter the address bus. Clock $\phi 1$ is also high when AEC goes high and freezes the buffers in the address line. When $\phi 1$ goes low, $\phi 2$ goes high. During the $\phi 2$ high, note that the memory access time takes place.

it receives a machine language STORE instruction. It also understands that it must direct data to either its instruction register or data bus buffer when it receives a LOAD instruction. These two instructions are the most used members of the Instruction Set.

OTHER TIMING SIGNALS

The 8502 needs other byproduct signals from the original clock signal. There are five of these other signals. Four of them the 8502 is able to generate, and VIC makes up the fifth one. The first signal is an edge to trigger the data bus direction. It is the R/\bar{W} .

The R/\bar{W} is a steady state that the instruction decoder sends out. Its energy is from $\phi 1$ and $\phi 2$ driving pulses. The R/\bar{W} line is connected inside the 8502 to the data bus buffer and also to the output pin 39. When the line is high, the buffers are enabled to receive data from the data bus. If the line is low the buffers are made to transmit data to the data bus.

When you look at the R/\bar{W} condition with reference to $\phi 1$ and $\phi 2$, certain coincidences become apparent. The state remains steady as $\phi 1$ and $\phi 2$ begin a cycle. Both of their edges take place and the R/\bar{W} line remains steady. The state stays that way for a period of time called *read/write setup time*. This is the time required for the R/\bar{W} state to be setup and become very stable. That way when the state changes it won't shake up everything.

The read/write setup time is calculated to be about 100 ns. At the end of the setup time, the R/\bar{W} is able to safely change states. The setup time begins as $\phi 1$'s high begins. The setup time finishes 100 ns into the $\phi 1$ high duration. If the 8502 is reading, then R/\bar{W} , at that time, goes high. Should the 8502 be writing, R/\bar{W} will go low.

Once the line goes high for a read or low for a write, it stays that way for awhile. It remains in that state for the rest of the $\phi 1$ high and during the $\phi 1$ falling edge. It continues to hold for the rest of the cycle as $\phi 1$ goes low. This is called, read/write hold time. The 8502 requires the R/\bar{W} to hold after the line stabilizes its state at least 10 ns, but it usually will hold for about 30 ns right at the end of

the cycle. After the hold time, the line could change. It is safe to make the change then. The rest of the sensitive lightning fast signals will not be disturbed.

Address Signals

The $\phi 1$ clock triggers off the address out of the PC. An address signal from the instruction decoder does the job. The address signal gets its start as the rising edge of $\phi 1$ goes from a low to a high at the beginning of the cycle. At that instant, the address setup time is begun. The signal travels to the program counter. The address setup time is typically about 100 ns. At the end of the setup time, the address signal changes states. The setup time is also needed with the address line in order not to destabilize the activity.

After the address signal changes states, the PC can then safely put the address bits on the address bus. The address signal then holds the state till the cycle is over. The address must be stable on the address bus at the end of the cycle for an address hold time of at least 10 ns. Actually the address bus will be holding for about 30 ns.

Another address signal in the C128 comes from VIC. VIC is considered in arcade game machines as the processor. In the C128 it is the 40 column video output, but it does exhibit some processor qualities. It is constantly addressing some sections of RAM for update information to keep the display intact on the TV screen. It shares the address bus with the 8502.

VIC does this by using one of its own generated signals, the address enable control, or AEC. The AEC line runs all around the C128. There will be more about the AEC connections to other chips in Chapter 20.

One of the AEC connections is at pin 5 of the 8502. Pin 5 is the input to the three-state buffers that can turn the address circuits in the 8502 off and on. In this way VIC can turn the 8502 off and on. VIC does what VIC can turn the 8502 off while it is running the C128. VIC turns on the 8502 when it relinquishes control.

The 8502, though, is not helpless. It can control things too. The 8502 must make sure that the

AEC is disabled and won't interfere while the 8502 conducts its business. The 8502 knows that the only time the AEC can shut it down is when the AEC line is low. When the AEC is high the 8502 has complete control. Therefore the 8502 conducts its affairs while AEC is high.

The 8502 must also observe an AEC setup time if the AEC is changing from low to high. The AEC setup time is a maximum of 75 ns. The AEC setup time must be concluded by the time the R/W and address signals have finished their setup time. There is no need for any AEC hold time. While the 8502 is operating the AEC line just stays high and does not interfere.

Data Timing

The R/W, address signal, and AEC all were timed with the high of the $\phi 1$ signal. Data timing works with the high of $\phi 2$ signal that occurs about 90 degrees after the $\phi 1$ in the same cycle. The first thing to observe in a processor is the memory read access time. The 8502 has an access time of about 300 ns. What does this mean? It refers to the amount of time that is available during a cycle to receive data from residents of the memory map.

The access time of the 8502 began near the finish of the high of $\phi 1$. As the read/write setup time ends, both R/W and the address signal have just experienced their rising edge. The memory read access time shown in Fig. 17-8 begins at that instant. The 8502 from that point in time has 300 ns to either read data from memory or write data to memory.

The 4164 DRAM chips must be able to send or receive data in well under the 300 ns access time of the 8502. If they have an access time of 150 ns or thereabouts, then they can easily be accessed during the 8502 accessing period.

Once the R/W and address signals are active, the access time of the 8502 commences. $\phi 1$ completes its high and goes through its falling edge. $\phi 2$ then begins its rising edge. In Fig. 17-8 it is shown that the trigger effect of the edge enables the addressed chip, and the data concerned is placed on

the data bus. During a read the data heads for the 8502.

It takes the data about 100 ns to make the trip from RAM to the 8502. This is sort of a setup time period. Once the data arrives at the 8502, it can enter one of two places. If the data is an instruction code, it is accepted by the Instruction Register. Should the data be an operand it is shuttled over to the data bus buffer. The layout of the program lines dictates which goes where.

Before the IR or data buffer will accept the bits, they must wait a short period of time while they stabilize. The trip from memory to processor could have upset the decorum of the data. The data stability time is about 60 ns. During stabilizing time the falling edge of $\phi 2$ occurs. This falling edge triggers the opening of the IR and the data buffer. The stabilized data is then allowed to enter the processor registers.

Once the data is strobed into its assigned register, a hold time must take place. The hold time is brief, about 10 ns. The data is then finally ready to be processed in the 8502.

During a write operation the routine is very similar to the read. The processor access time is the same 300 ns and begins in the same way, that is when R/W and the address signal finish their beginning edge. The $\phi 1$ high then falls and the $\phi 2$ rising edge takes place. The memory address is enabled and the 8502 places data on the data bus. The data then speeds to the addressed location. A data setup time takes place for the write operation. It is about 100 ns. The data from the 8502 approaches its memory destination.

Again there is a data stability time segment of about 60 ns. The falling edge of $\phi 2$ occurs. The stable data is then let into the location and the bits are stored in their respective bit holders. A 10 ns hold time takes place. The transfer is complete.

Reading and Writing to Peripherals

When the 8502 wants to read data from a peripheral, all it has to worry about is the setup time. The R/W and the address signals arrange the direc-

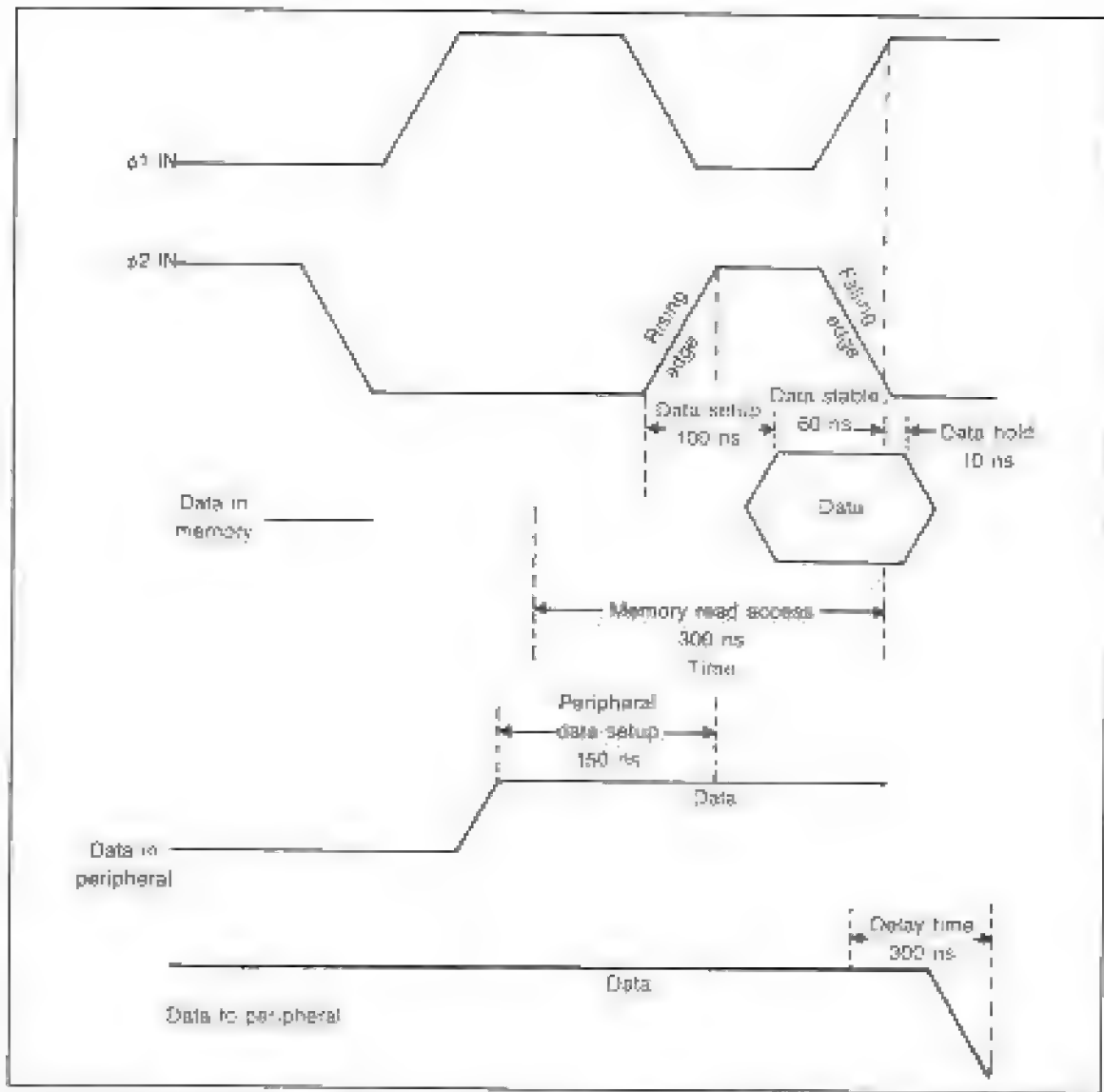


Fig. 17-8. Data from memory must be stable and valid when the falling edge of $\phi 2$ strobes it into the 8502. The data needs a 100 ns setup time and a final 10 ns hold time.

tion of the data bus and open up the register in the peripheral. This is all done during the rising edge and the high of $\phi 1$.

A setup time of about 150 ns is then needed to stabilize the data bus to receive the data. Again refer to Fig. 17-8. Once the setup time has elapsed, the rising edge of $\phi 2$ occurs and the data is placed

on the bus. The data travels to the 8502 and the high of $\phi 2$ takes place. Then the falling edge of $\phi 2$ copies along. The data is given entry to the 8502's IR or data buffer for processing.

The writing to peripherals is the reverse operation. The 8502 addresses the peripheral and makes the R/W low for a write. As the $\phi 2$ rising edge com-

times, the 8502 places the data for the peripheral on the data bus. The data then takes a trip to the peripheral. The falling edge of $\phi 2$ occurs and the data is strobed into the peripheral register. A setup time was not needed, but a considerable hold is required. It is called delay time. The data needs a 300 ns delay in order to ensure that the data to the peripheral arrives with validity. This works fine, except that it slows down the transfer of data to the peripheral.

The way the clock marches bits to address locations and transfer data has a lot of intricate steps. Each step is simple and not confusing. There are just a lot of steps.

The 2 MHz Mode

The so called 2 MHz clock has two modes. The 1 MHz mode of the 2 MHz clock has just been discussed. What about its 2 MHz mode, the one the clock is named for? The first thing to consider is that the clock is running twice as fast. In the 2 MHz mode, the clock is completing two million cycles every second. This change cuts the number of nanoseconds the clock takes for a cycle to about half of the ns needed for the 1 MHz mode. While the 1 MHz mode needed about 1000 ns to cycle, the 2 MHz mode needs about 500 ns. According to the specification sheet, the 2 MHz clock is said to take exactly 489 ns. All the other timings such as setup times, hold times, rising edge times, falling edge times and so forth, are mostly cut in half too. Table 17-1 shows the timings an 8502 in 2 MHz mode requires.

Table 17-1. With a 2 MHz clock some of the timings are halved compared to the 1 MHz version.

TIMINGS FOR 2 MHz CLOCK	
Timings	Nanoseconds
Cycle Time	489
Rising Edges	10
Falling Edges	10
Address Setup	50
AEC Setup	37
Address Hold	40
Data Setup	100
Data Stable	40
Data Hold	40

A complication can happen when the 8502 is made to run at 2 MHz. The 2 MHz works smoothly when the 8502 is transferring data to and from the RAM, ROM and other internal chips. When the 8502 is performing I/O operations, however, the clock speed in 2 MHz mode sometimes must be adjusted to the 1 MHz mode. The 1 MHz clock drives all the I/O chips even when the 8502 is in 2 MHz mode.

This adjustment is called clock stretching. Figure 17-9 shows the stretching technique in 2 MHz mode. The top square wave shape shows one 1 MHz cycle. During one 1 MHz cycle, there are two 2 MHz cycles. Note both a low and a high in the 2 MHz waveshape during the low of the 1 MHz waveshape. Then there is another low and high during the high of the 1 MHz wave. There are two complete 2 MHz cycles during one complete 1 MHz cycle.

In the second set of waves in Fig. 17-9, the top wave shows the 8502 making an I/O access in the 1 MHz mode. This is the normal wave arrangement the 8502 uses to access I/O. It works perfectly without complication. The third set of waves shows the way the 2 MHz mode must be stretched in order to adjust to the 1 MHz I/O access wave. The stretching is performed by a concerted effort between the 8502, VIC and the PLA. If they didn't stretch the clocking, the I/O access would be out of sync with the 1 MHz clock that is driving all the I/O chips.

THE Z80 TIMING

The Z80 is in strange territory in the C128. The circuits in the C128 are all designed to support the 8502. The Z80 must be equipped with U12 and U13 in order for it to adjust to the 8502 circuitry. The Z80 uses U12 and U13 to latch and buffer its way in and out of the data bus.

The 8502 converts its clock input from VIC into two individual phases, $\phi 1$ and $\phi 2$. The Z80 on the other hand receives its own Z80 clock from VIC but does not split it into two phases. The Z80 has a single phase clock. That is the big difference between the Z80 and the dual phase 8502. The Z80 goes right ahead and uses the clock as is.

VIC supplies a clock that beats at a 4 MHz frequency. That makes the clock twice as fast as the

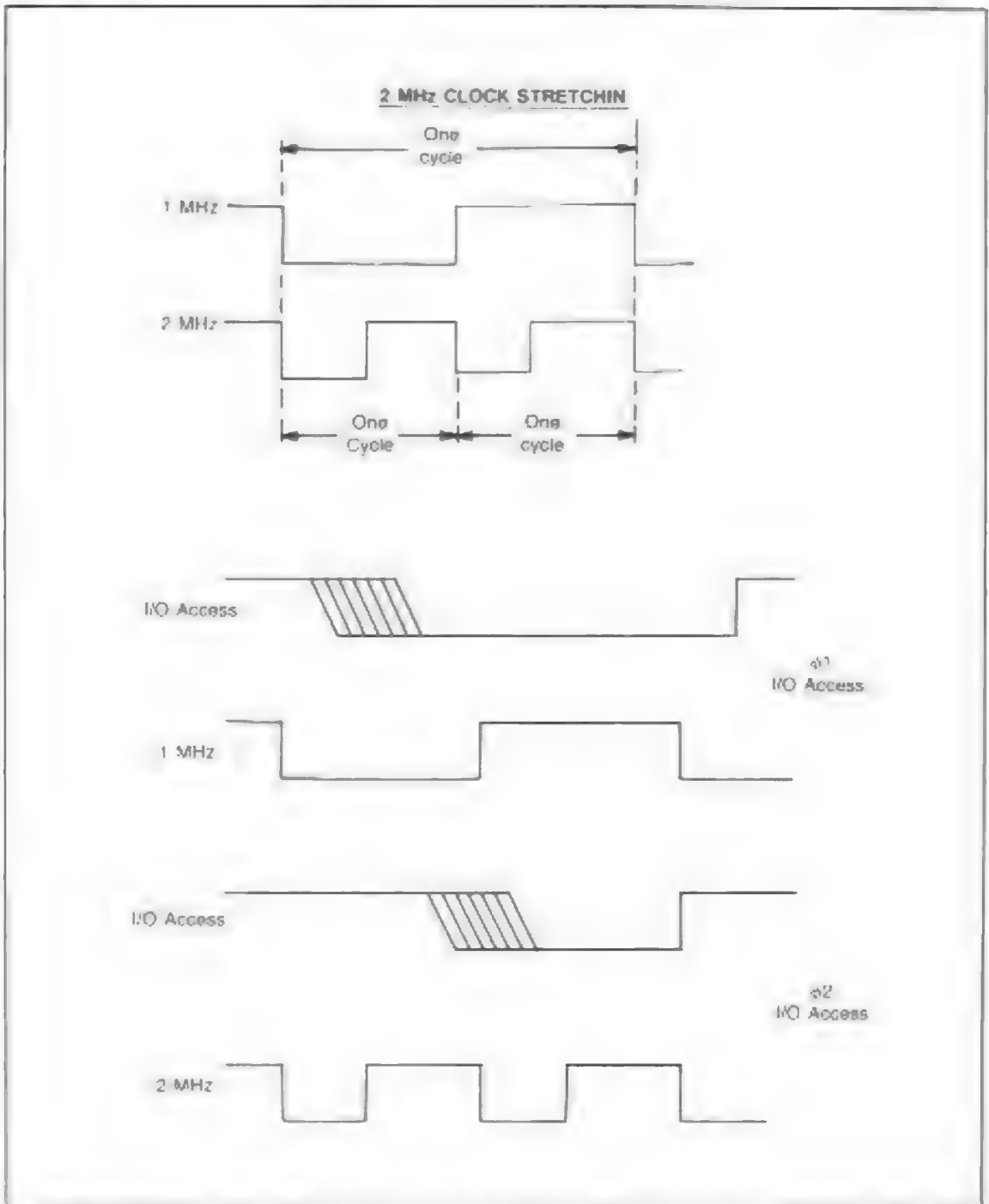


Fig. 17-9 The 2 MHz clock must be stretched in order to compensate for the fact that the I/O can only operate in the 1 MHz mode

2 MHz and four times faster than the 1 MHz. However, the VIC only supplies clock pulses during AEC low. When AEC is high, VIC shuts down the Z80 clock output. This makes the clock wave train consist of two fast 4 MHz pulses during AEC low, then no pulses, just a continuous low during AEC high and then two more pulses as AEC goes low again. This is shown in Fig. 17-10. The Z80 is only connected to the system bus line during AEC low. During AEC high it is disconnected. The 8502 on the other hand is connected to the bus lines during AEC high and disconnected during AEC low, when the Z80 can take over. This is the basis for the two processors to be able to work together, switching from one to the other.

With the processor running at 4 MHz but only being on during AEC low, it can run the CP/M operating system at an effective 2 MHz rate. The Z80 acts as if a continuous 2 MHz clock is doing the bit driving. Inside the Z80 the clock is called ϕ (phase). A square wave high and low is called, one clock period. An instruction cycle is composed of about 12 clock periods. The 12 clock periods are the op code fetch, the memory read and the memory write, as seen in Fig. 17-11. In Fig. 17-12 and Fig. 17-13 the timings are shown.

The rising and falling edges trigger the circuits in the processor. The first rising edge in the op-code-fetch group triggers the program counter to place its address bits onto the address bus. The first falling edge causes pin 19, \cdot MREQ, and pin 21,

\cdot RD, to output bits to the memory to produce a read. This causes a copy of the stored op code onto the data bus.

The second clock pulse marks time. The rising edge of the third clock strobes the op code that was fetched, into the instruction register of the Z80. That third rising edge also turns off \cdot MREQ and \cdot RD.

The memory write operation is quite like the fetch or read operation. The first rising edge places the address bits on the bus in the same way. It also triggers \cdot MREQ and \cdot RD on their way. What is \cdot RD doing in this write operation? If you look at the timing diagram, Fig. 17-13, you'll see the \cdot RD signal is low during the memory read clock periods but goes high and not in operation as the memory write clock periods take place. As the clock gets into memory write time, the \cdot MREQ goes high too but then goes low as the falling edge of the first write clock takes place.

As the rising edge of the first write clock begins, pin 22, \cdot WR, outputs a high. Then as the fall of the second write clock edge takes place, it makes \cdot WR go low. This in turn causes the data from the processor to traverse the data bus and be strobed into the addressed location.

That is the way the clock periods are used in single phase clocks to fetch instructions, read data, and write data. There are also a lot of other basic duties the clock is made to perform. It must energize the reading and writing to I/O chips, the refreshing of dynamic memory, the fabrication of the

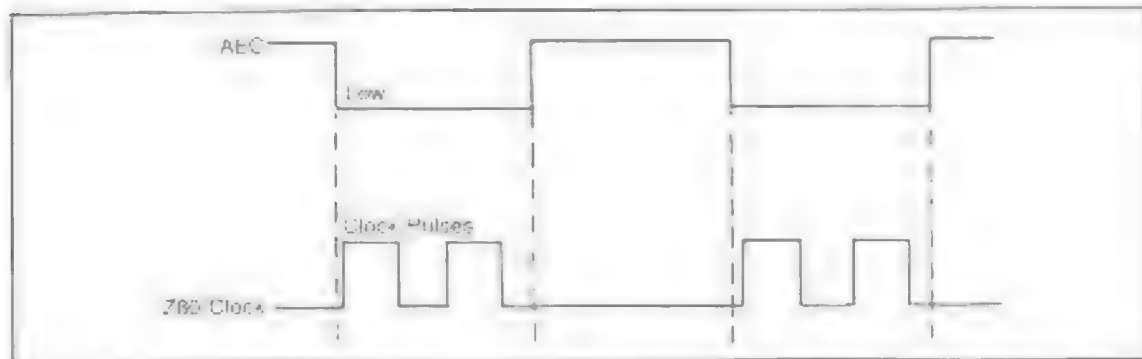


Fig. 17-10. In order to run, the Z80 VIC must supply a 4 MHz clock because VIC only supplies the clock during AEC low. This allows the Z80 to operate at an effective 2 MHz rate.

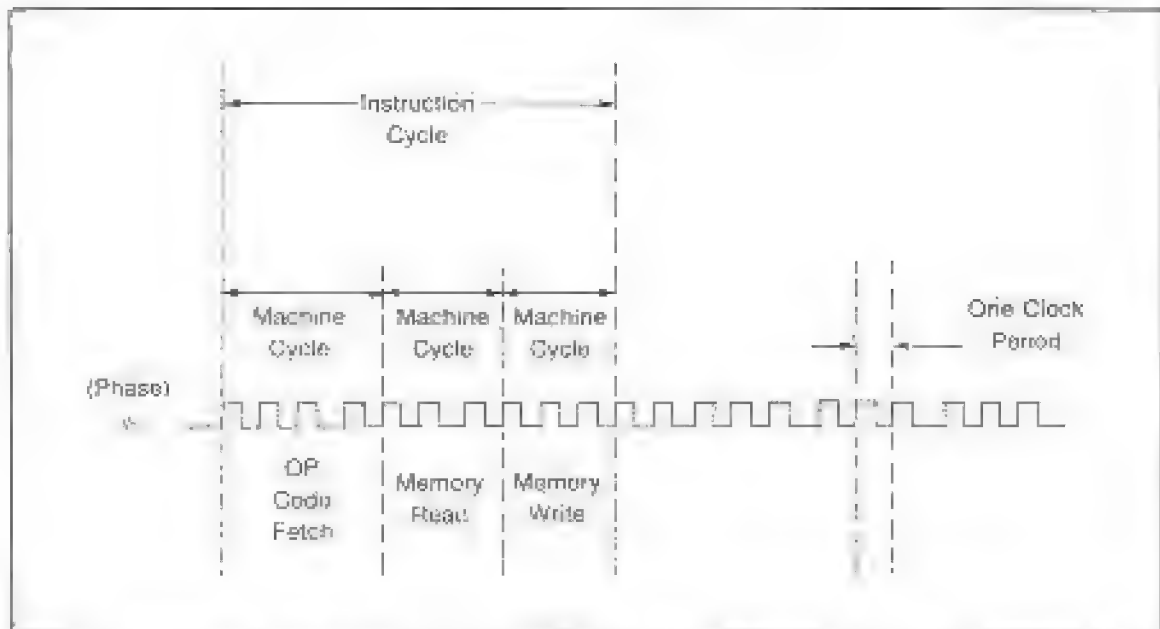


Fig. 17-11. The Z80 uses a single phase clock. Each clock period is one high and low. About a dozen clock periods are in an Instruction Cycle. The Instruction Cycle has three machine cycles. They are the Op Code Fetch, the Memory Read and the Memory Write.

composite TV display signal, production of audio tones and other things. All the bits in the U128 march to the beat of the clock.

TESTING THE CLOCK

When the computer is receiving power and is not producing any output at all, the clock could have stopped. It is useful then to be able to quick-check the clock to see if it is running or not. The best place to check it is at a stage after the sine wave has been converted to a square wave. If you test near the crystal, the test probe could load down the oscillator circuit and kill the frequency even if the clock is indeed running.

In the U128, a convenient place to test is the 2 MHz input at pin 1 of the 8502. It should read PULSE on the logic probe if the clock is running. Figure 17-14 shows what the pulse should look like if you view it on an expensive scope set at 0.2 microseconds. The waveshape will have a peak-to-peak voltage of about 5 volts.

If you have just an ordinary TV repair scope, then you can't view the 2 MHz frequency, but touching down on the pin will let you see an envelope containing the frequency, as in Fig. 17-15. Even though you can't read the frequency, if the envelope does appear, then the clock is running. The clock frequency also is probably okay because the crystal cut for 14.31818 MHz won't ring unless the master frequency is near that value.

If you are only equipped with the logic probe, there are many test points that will verify that the clock is on. Any of the address or data bus connections should show PULSE if the clock is on. In fact, any of the pins on all the Test Point Charts that indicate the presence of a pulse are clock test points. If a pulse is at any of these pins, the clock is oscillating. All of these pulses are originated in the clock chip. Of course, if there are no pulses anywhere in the machine, the clock has been stopped.

A vwm with a needle indicator, not a digital vwm, can also provide a quick go/no-go test of the clock. When you touch down on a pulse test point the bee-

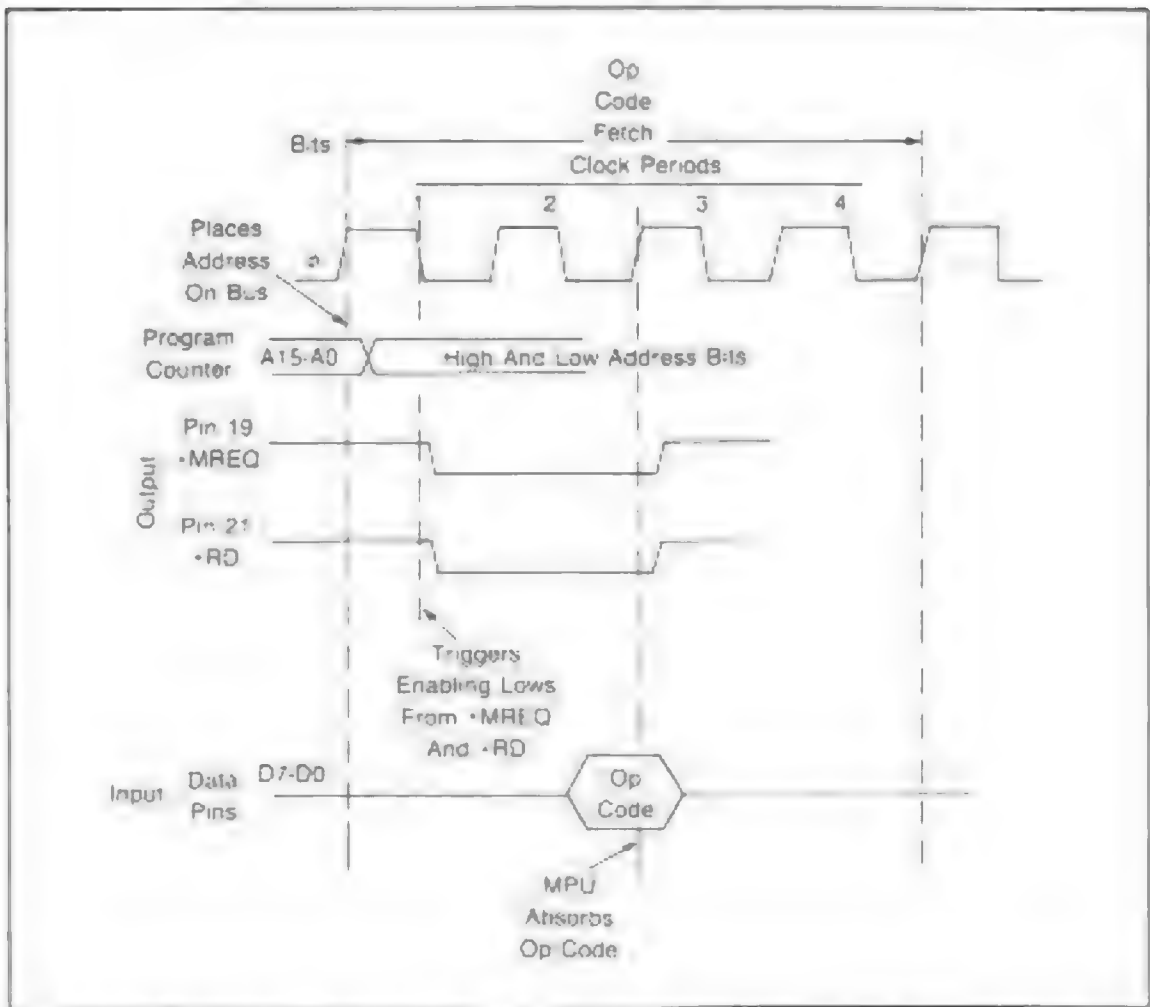


Fig. 17-12 The first rising edge of the fetch triggers the program counter to place the address bits on the address lines. The first falling edge triggers $\overline{\text{MREQ}}$ to enable the memory chips and enables the $\overline{\text{RD}}$ signal to cause a read. The op code enters the data bus. The third rising edge strobes the op code from the data bus into the Z80. The third rising edge also turns off $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$.

die will read a sort of three-state value around 2 volts on the meter face. The needle will appear a bit unstable and might show an appreciable wobble. This means a running clock.

It was mentioned earlier that a frequency counter and an expensive scope could be useful. The frequency counter could provide you with the actual frequency at which the clock is running. You can ad-

just the trimmer capacitor C20 for a frequency of 14.31818 MHz. The counter should be connected to pin 8 of U28, the 8701 clock chip.

If you decide to perform this adjustment, then try to use a pickup loop rather than a direct connection. The loop will pick up the frequency by induction and not make a physical connection to the clock circuit. This is advisable if possible. If it is not easy

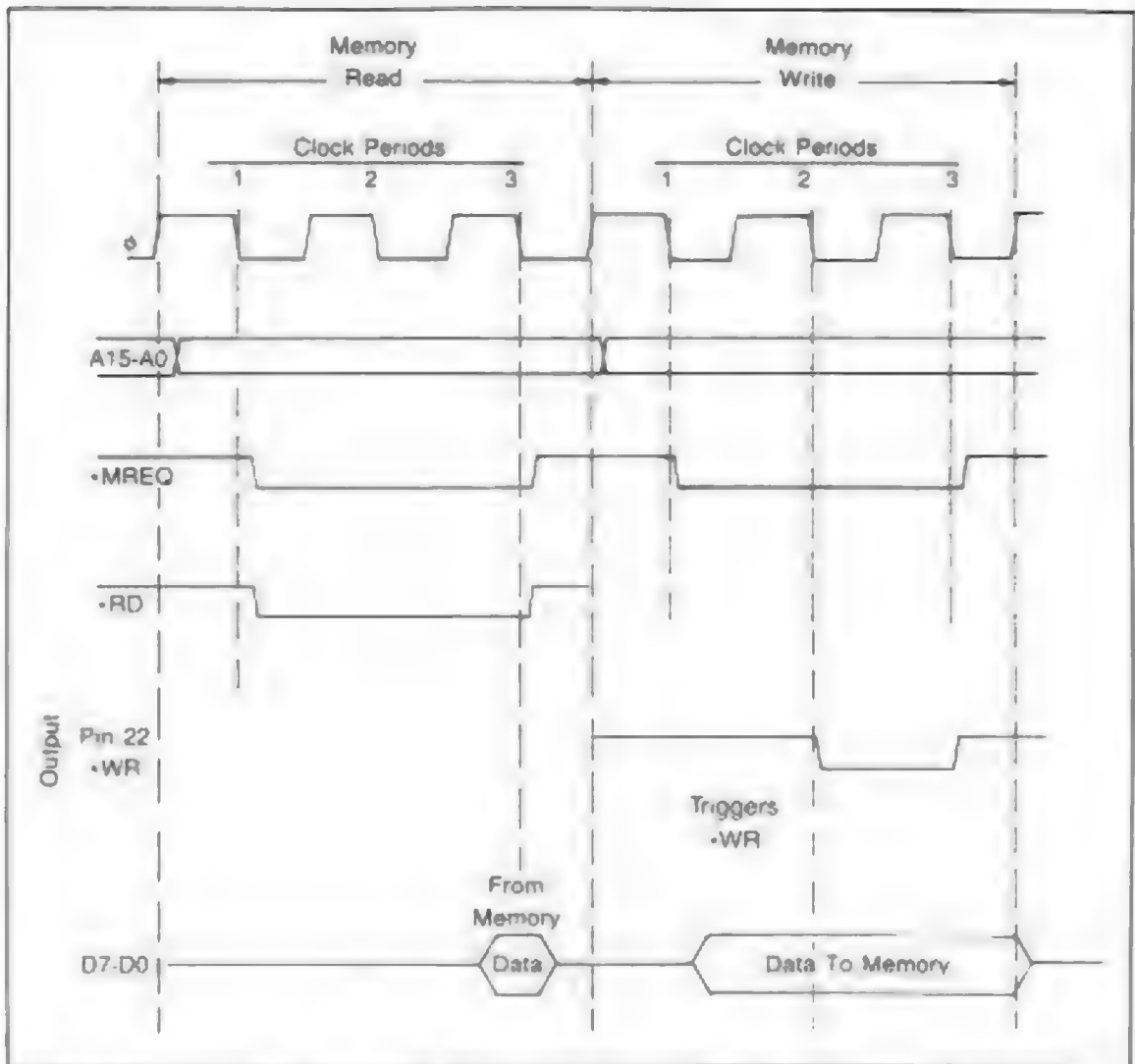


Fig. 17-13. The memory read operation is very similar to the fetch operation. The rising edge places the address on the bus. The falling edge triggers $\cdot\text{MREQ}$ and $\cdot\text{RD}$. The falling edge of the third clock strobes the data into the Z80. The memory write operation though is somewhat different. The addressing is the same, takes place during the rise of the first write clock, and $\cdot\text{MREQ}$ is enabled at the fall of the first clock. $\cdot\text{RD}$ is not used, but $\cdot\text{WR}$ is. Data enters the data bus and strobes. At the fall of the second write clock, $\cdot\text{WR}$ strobes the data into the memory location.

to do then use the direct connection—just be careful making the connection. You could possibly load the circuit, stop the clock and get misleading results.

An expensive scope will actually display the various clock waveshapes and frequencies. It is rare,

however, that you'll need such scope details for repair work. These waveforms are a lot more valuable to the design and manufacturing engineers rather than to a field servicer. Figure 17-16 is the Test Point Chart for the 8701 clock chip.

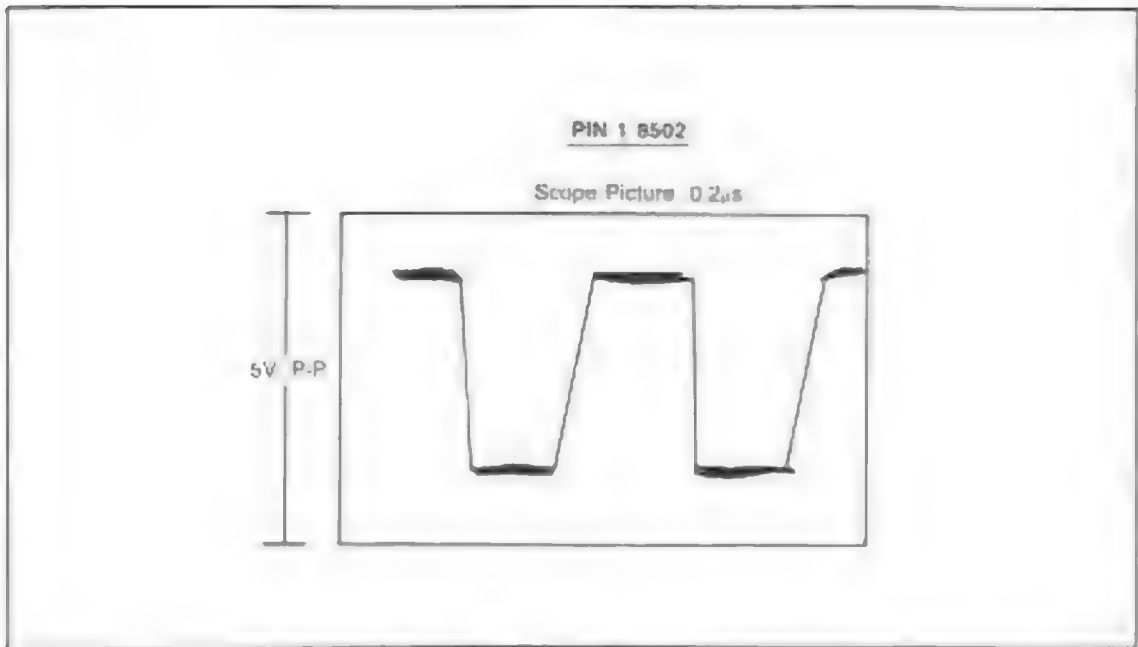


Fig. 17-14. If you test for the 2 MHz clock at the B502 with an expensive scope at 0.2 ns, you'll see a couple of square waves

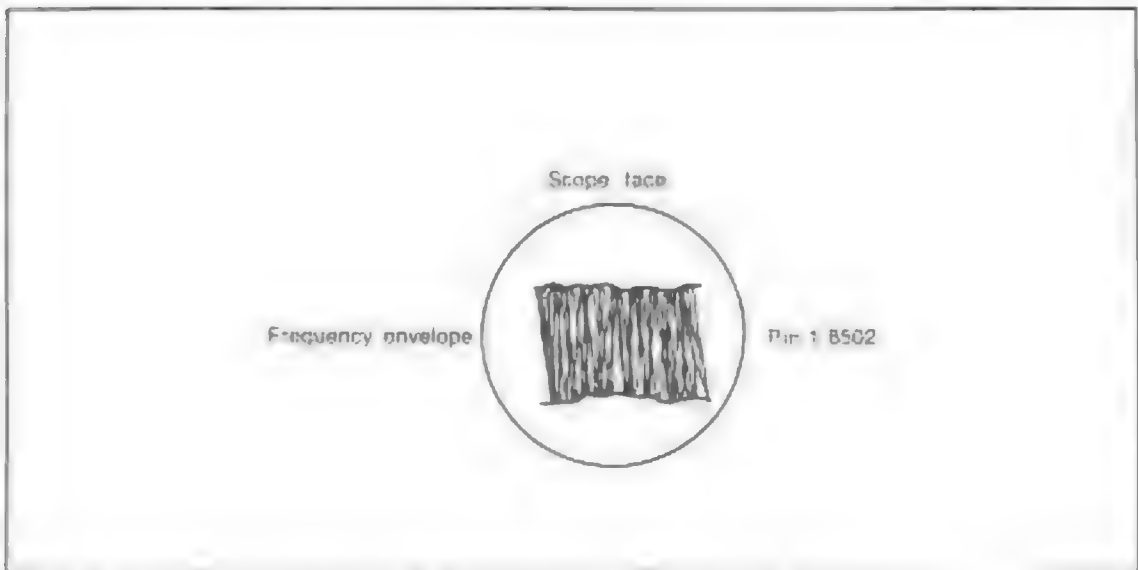


Fig. 17-15. An ordinary TV repair scope will display an envelope containing a lot of waveshapes

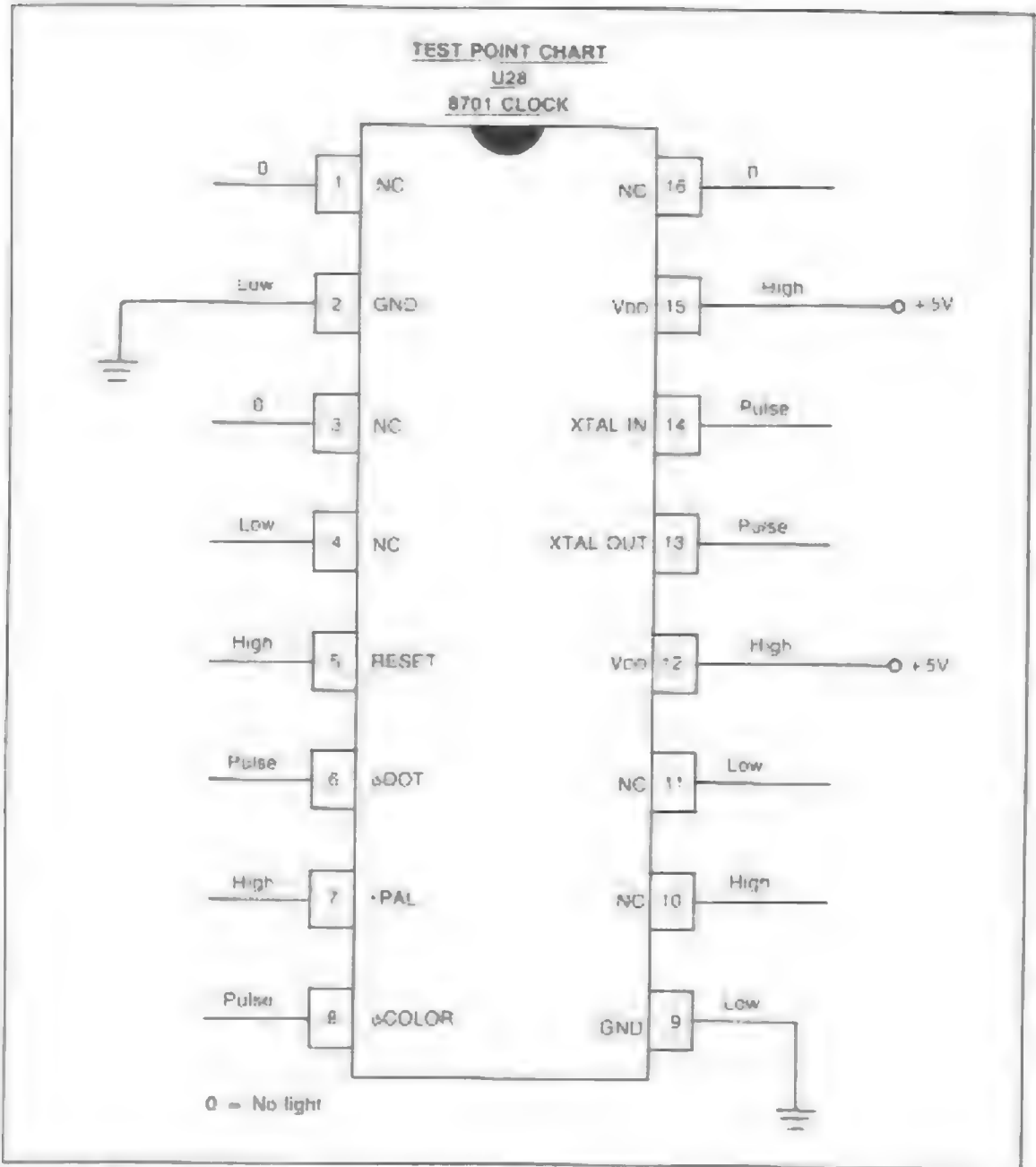


Fig. 17-16 The logic probe and the Test Point Chart will tell you if the clock chip is running properly.

18. Address, Data and Control Buses

The C128, the C64 and the Z80 computers, all encapsulated onto the single printboard, share three types of bus lines to pass bits between themselves and the chips with which they communicate. For addressing purposes, they all use the 16-bit address bus. To pass bytes of data back and forth they all use the eight-bit two-way data bus. For the many ways they need to control the circuits, there are a number of input-only, output-only and two-way individual control lines.

All of the bus lines transfer digital highs, lows and pulses. There are no other types of signals in the digital circuits. In the port circuit areas, there can be other signals such as audio and color video, but in the digital circuits there are only highs, lows, pulses or turned off three-stated voltages. No other working condition can be attained. If you are testing the lines, then those are the states you are looking for. These copper traces that travel around the printboard and connect to all the major chips are very vulnerable when the board is exposed. They are of-

ten the source of problems during repair jobs. These troubles are covered in this chapter.

ALL THOSE ADDRESS BUSES

The address buses, Fig. 18-1, originate in the 8502 and Z80. Coming out of pins 7-23, except for pin 21 (GND), are the A0-A15 lines from the 8502 program counter circuits. From pins 30-40 and 1-5 of the Z80 are the A0-A15 lines from its program counter. These address lines are joined together into one bus. The 8502 and Z80 can easily share these address lines because they are both never on at the same time. Either the 8502 has control or the Z80 is in charge but never both together.

Just inside both of the processors, between the pins and the program counter, are 16 three-state buffers. The buffers can be turned on and off as needed by an input control line from VIC. VIC is a processor in its own right. It produces AEC, address enable control. AEC is very important in controlling these address buses and will be discussed in more

detail later in this chapter. It does a lot of turning off and turning on as required.

Once the lines arrive at the output pins of the processor chips, they assume the names A0-A15. Some or all of the lines connect to chips on the memory map. Each chip and every register on each chip is given its own address. Each individual address can be opened up while all the rest of the addresses remain closed. The 16 lines are, as mentioned many times, able to carry 65,536 individual sets of highs and lows. These sets of highs and lows are each a combination to open up one and only one address. If more than one address opens during an addressing operation, then the program could crash.

In this machine, however, multiple floors are in the many-storied memory map building as shown in Chapter 16. Many registers are on different chips with the same address. The map is a many-storied building with the same addresses on each floor. Each floor has a possible 64K addresses. A bank picks and chooses chips from the eight floors to form a 64K block. A bank number must be added to the 16-bit addresses to reach the chosen block.

During troubleshooting, PEEK and POKE in BASIC or MEMORY and FILL commands in machine language can be used to good advantage. These troubleshooting signal tracing and injection techniques operate over the address and data bus lines. As a result you can use them to test these bus lines. The actual techniques are discussed at the end of this chapter.

The A0-A15 bus that exits both the 8502 and Z80 are connected together line for line. These lines go directly to the RAM multiplex chips, ROMs and CIAs. They also connect to the MMU, the PLA, the 8563 Video Controller and SID.

Translated Address Bus

All 16 address lines connect to the MMU. The MMU processes them and then outputs the translated address bus. Only eight lines are in the translated address bus, TA8-TA15. These lines are generated from the input lines A8-A15. The trans-

lated address bus performs addressing chores for RAM and VIC.

First of all, lines TA8-TA15 are connected to the two multiplex chips (U14 and U15) as shown in Fig. 18-2. TA12-TA15 connect to U14 and TA8-TA11 to U15. They represent the high order address bits. The regular low address lines, A0-A7, are also connected to U14 and U15: A4-A7 to U14 and A0-A3 to U15. With these inputs the 74LS257 multiplex chips are able to form the RAM row and column addresses.

The next job the translated address bus does is use its TA12 line to substitute for A12 in addressing ROM4. ROM4 is the ROM that must get its address changed when the Z80 takes over. If you recall, ROM4 gets its hex addresses D000-FFFF changed to 0000-000F. TA12 from the MMU performs this task. The translated address bus also is used in C64 mode. It becomes address lines 8-15 of the expansion port.

The translated address lines are also involved during the time that VIC is in control and doing some addressing. When VIC is in charge, the MMU tristates TA8-TA11. Then it pulls TA12-TA15 high to get them out of the way. VIC is then able to use the TA8-TA12 lines to output its VIC addresses VA8-VA11. More about this is given in the next section.

When VIC is no longer in charge but an external device is in charge such as a cartridge, all of the MMU TA lines are tristated, leaving the lines free for use by the external device. The device can then use the regular address lines, A8-A15. The device can then address the entire memory map except for the MMU.

The Two Multiplexed Address Buses

The two multiplex chips mentioned above, U14 and U15, have inputs A0-A7 and TA8-TA15. The two chips also connect to VMA0-VMA7—the VIC Multiplexed Address Bus. These same signals are also passed through 33 ohm resistors. Once the signal passes through the resistor, this output set of

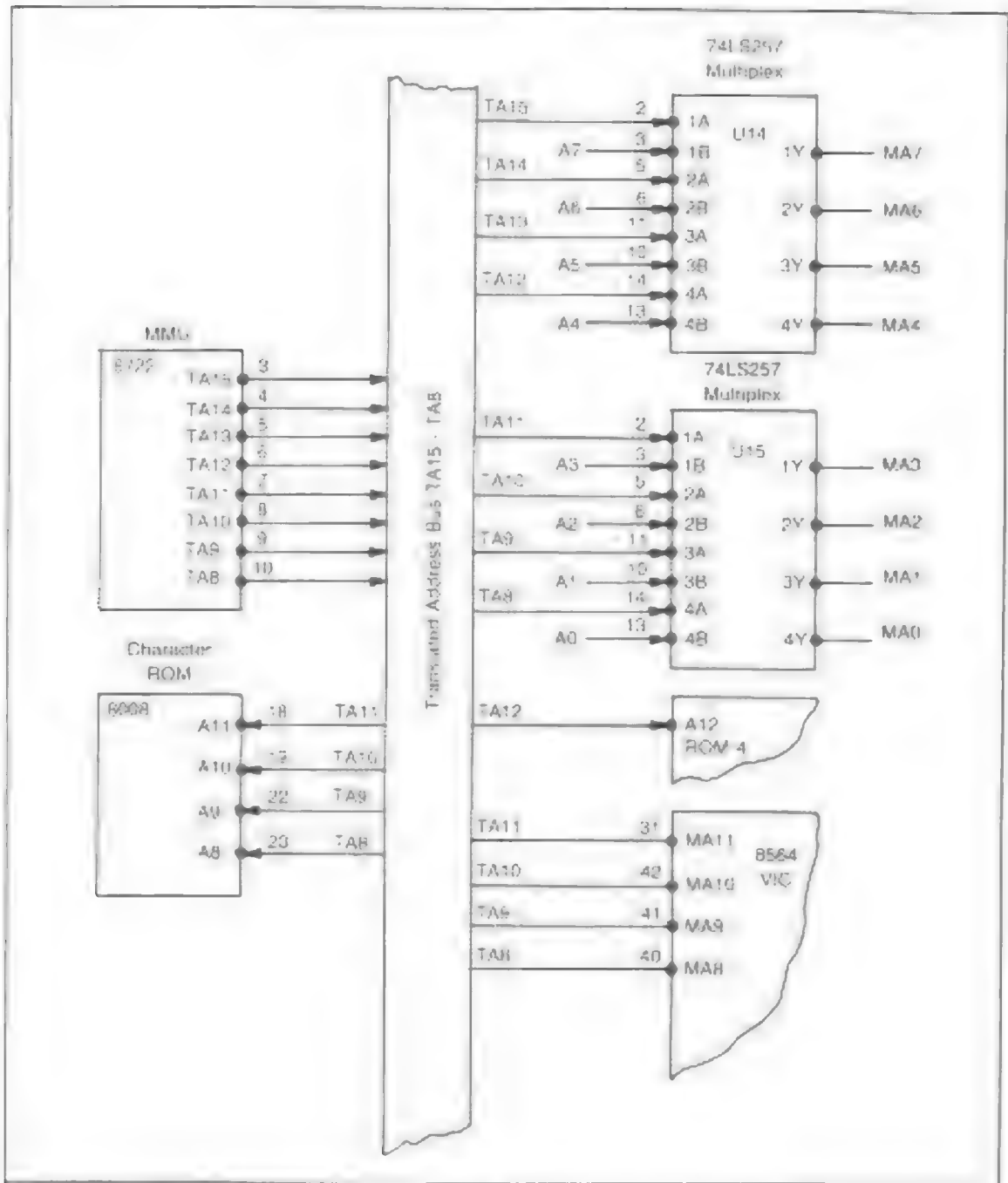


Fig. 1B-2 The Translated Address Bus is generated in the MMU. It contains bits TA15-TA8

bits is named: MA0-MA7, the Multiplexed Address Bus. The connections of the bus signals are illustrated in Fig. 18-3.

The VMA0-VMA7 and MA0-MA7 are produced by U14 and U15 when AEC, inserted at pins 15 on the two chips, is high. When AEC is high the two chips multiplex TA8-TA15 with A0-A7. The control for the multiplexing comes in pins 1 with the MUX signal. The 33 ohm series resistors then take the VMA0-VMA7 and, in turn, multiplex them to produce MA0-MA7.

The VMA signals are connected to VIC. The MA signals are sent to the system DRAMs. The processors can then address the VIC registers or address the 128K RAM set.

These VMA lines are also used by VIC when it needs to access sections of RAM. When AEC goes low and VIC is in charge, VMA becomes an output address bus from VIC. VIC uses VMA0-VMA7 plus the other address lines that are all three-stated during the AEC low.

More detail is given on how VIC accesses and refreshes the DRAMs in Chapter 20. To summarize, VIC only uses 14 address lines, as shown in Fig. 18-4, not 16. It is only required to address 16K directly. VIC receives VA15 and VA14 from CIA2, discussed in the next chapter. VIC generates the output bits over VMA0-VMA7 for the DRAM accessing and refreshing. The other lines output by VIC, TA8-TA11, are used by the next address bus discussed, the Shared Address Bus. VIC accesses the character ROM and color RAM with these.

The Shared Address Bus

The Shared Address Bus is called "shared" because both the processors and VIC use it when they are in charge. It is not multiplexed. The Shared Address Bus is needed to allow both the processors and VIC to address the character ROM and the color RAM. Both of them must use these two chips and they are approaching the chips from different directions. When the processor accesses the chips, as in Fig. 18-5, it sends bits in one direction. When VIC is accessing, as in Fig. 18-6, the address bits are coming from the other direction.

When AEC is high and the processor is in charge, the Shared Address Bus, SA0-SA7, is an output of U55: the 74F245 Transceiver chip. A transceiver indicates two-way transmission. When AEC is high, address bits A0-A7 enter the chip and SA0-SA7 exit. The SA0-SA7 output is fed to the character ROM and the color RAM. The output of these chips are into the system data bus. The higher order address bits are supplied by TA8-TA11, another form of shared address bus. With this addressing mechanism the processors are well able to access the character ROM and the color RAM from its direction.

When AEC goes low, the VIC addresses VMA0-VMA7, also called VA0-VA7, then takes over the Shared Address Bus and proceeds to send bits over the bus from the opposite direction from which the processors sent bits. You could say the bits are driving the bus backwards.

VIC supplies the lower order bits, and the translated address bus higher order address bits. The character ROM needs TA8-TA11 while the color RAM only requires TA8 and TA9.

When a cartridge is in place, the shared address bus is able to supply the lower order of address bits for the Expansion Port. This permits VIC to address the cartridge. Should the cartridge be given charge of the system, the SA and TA lines can be driven backwards and address RAM and ROM.

To sum up: four address bus systems are in the C128. First are the regular A0-A15 address lines. Second is the Translated Address Bus, TA8-TA15, that the MMU generates. Third is the VIC Multiplexed Address Bus, VMA0-VMA7 and its cousin the Multiplexed Address Bus, MA0-MA7. These buses are used by VIC when it must do addressing. Lastly is the Shared Address Bus, SA0-SA7. This bus is needed so that both VIC and the processor can address the character ROM and the color RAM.

THE DATA BUS

In comparison to the address buses, the data bus is a simple affair. It has eight lines, D7-D0. Every register in the memory map, except for color RAM, is eight bits wide. Color RAM registers only

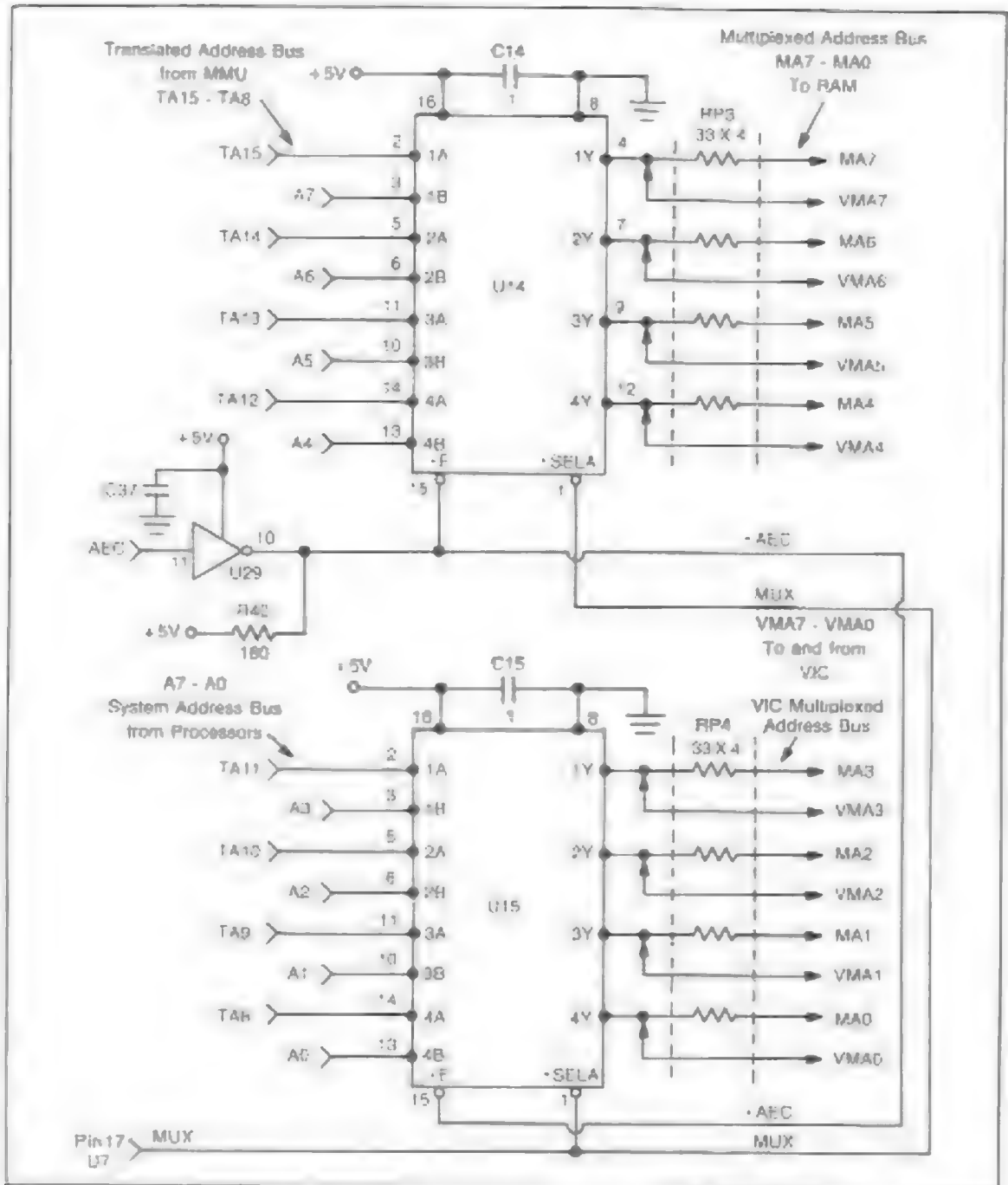


Fig. 18-3 The two multiplex chips U14 and U15 are the crossroads for the different address buses. They receive bits TA15-TA8 and bits A7-A0 from the MMU and the processor. This permits processor accessing of RAM and VIC. They also receive VMA7-VMA0 from VIC. This allows VIC to be able to access RAM. MA7-MA0 are the bits that access RAM.

VIC ADDRESSING RAM
16K ACCESS (14 BITS)

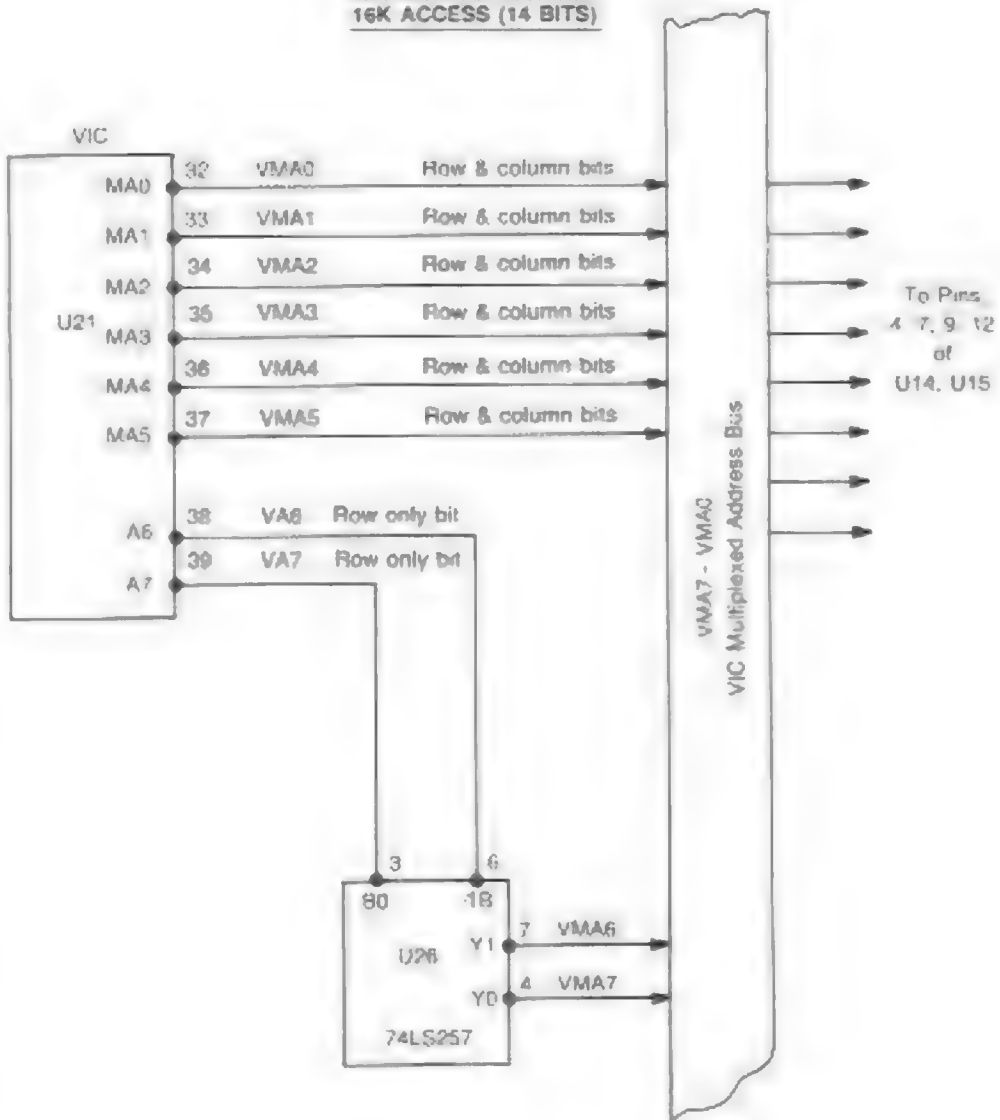


Fig. 18.4 VIC only uses 14 address lines to access a 16K section of RAM. It multiplexes eight row address bits and six column address bits.

PROCESSOR ACCESS

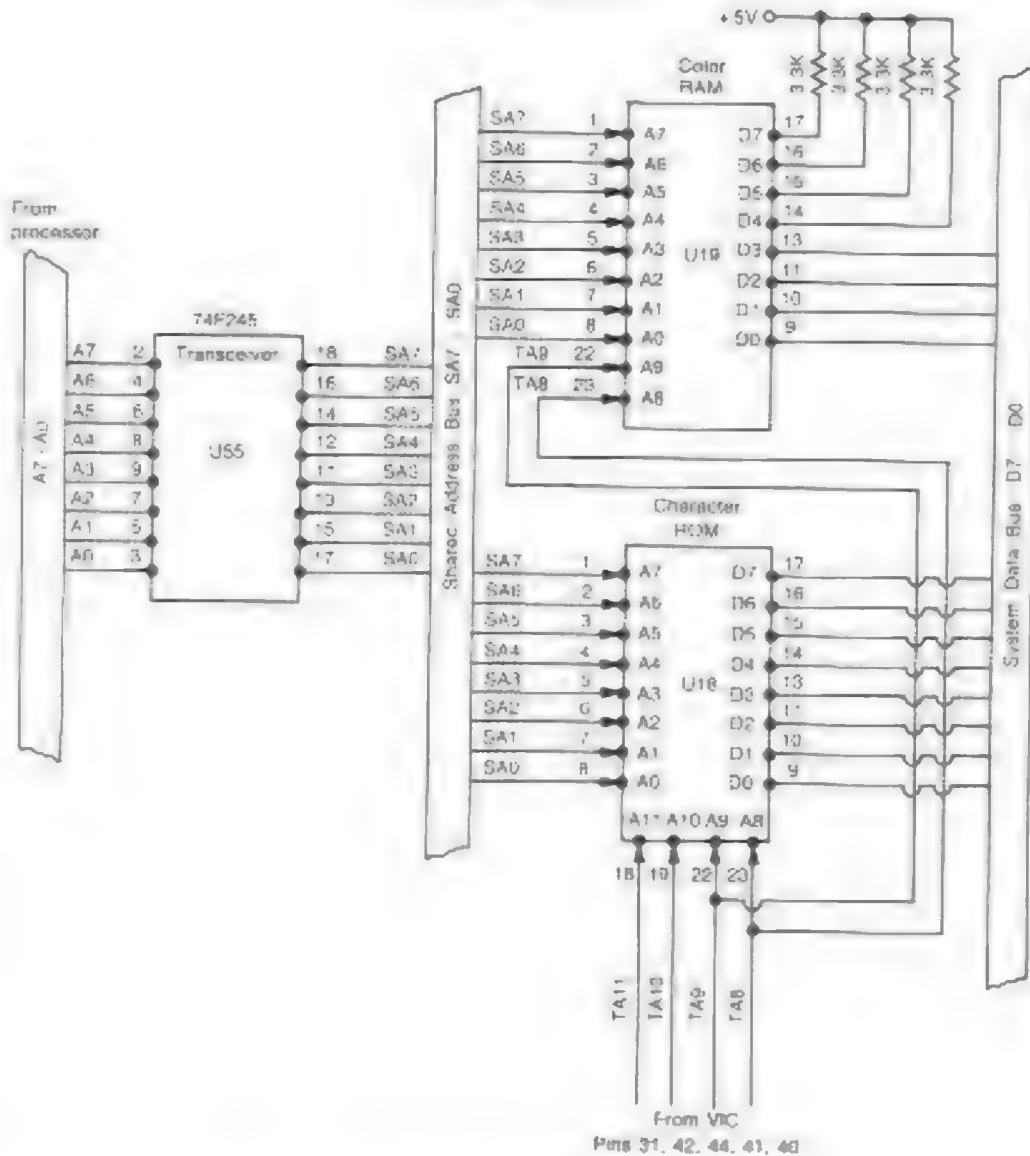


Fig 18-5. When the processor accesses the Character ROM or the Color RAM it uses the Shared address bus, bits SA7-SA0. The transceiver chip, U55, has bits A7-A0 enter it and outputs SA7-SA0.

VIC ACCESS

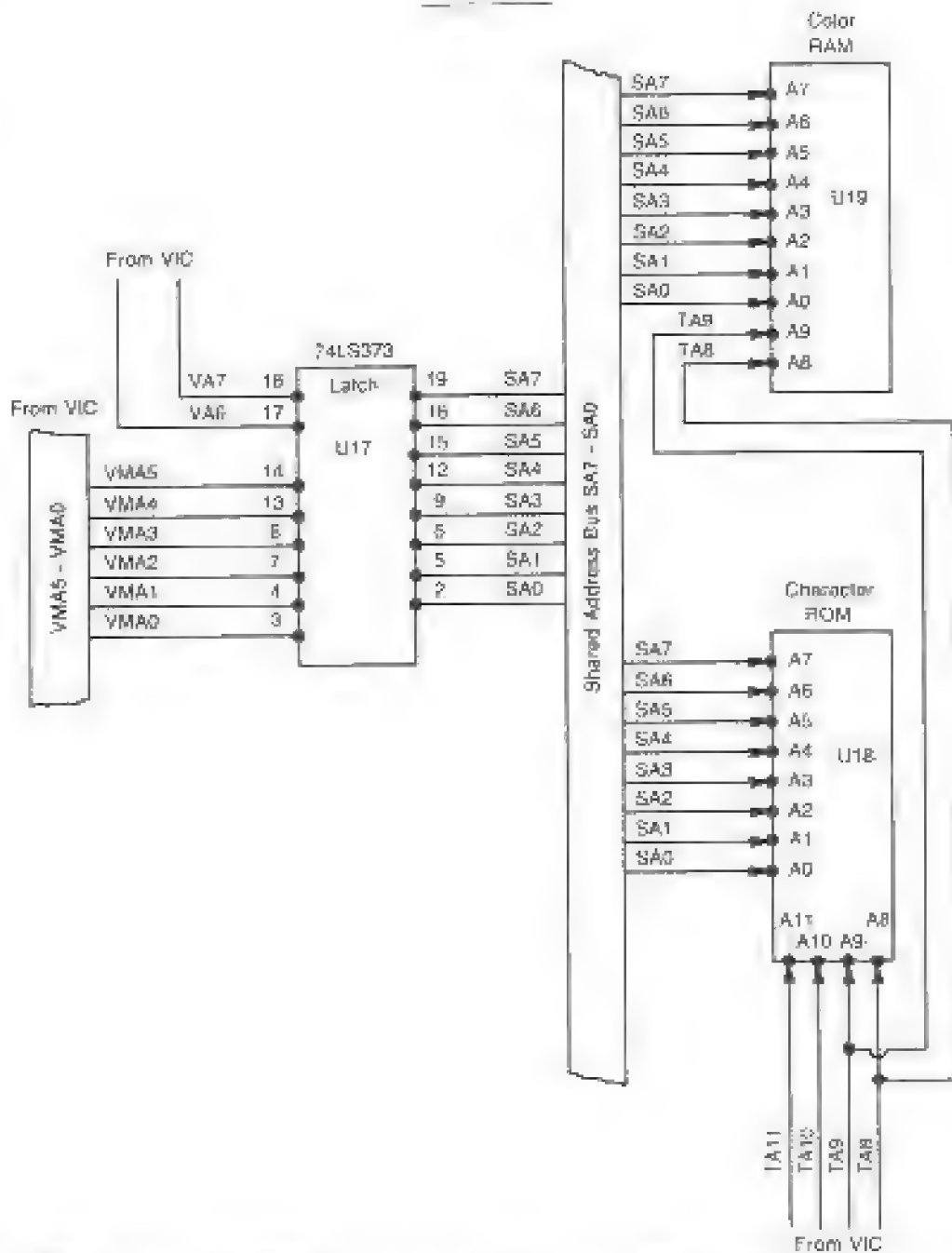


Fig. 18-5. When VIC accesses the Character ROM and the Color RAM, it inputs VIC multiplexed bits, VMA5-VMA0, VA6 and VA7 to the latch, U17. The latch outputs SA7-SA0 to produce the access.

use four of the eight bits in their registers, making the color RAM registers effectively only four bits wide. On the actual color RAM chip, U19, a 2016, eight bits are in each register. The eight bits attach to pins 9-17, D0-D7. Pins D4-D7, however, connect to four 3.3K resistors, that in turn are wired to +5 volts. This arrangement holds D4-D7 high and effectively out of the circuit, as in Fig. 18-7.

When the color RAM is addressed by the processor, or VIC, the lower four bits of the data bus handle the register accessing. The higher four bits of the data bus are ignored in the scheme of things.

The color RAM is addressed and a nybble of data is accessed by the processor when AEC is high. The access takes place through U20, the 4066

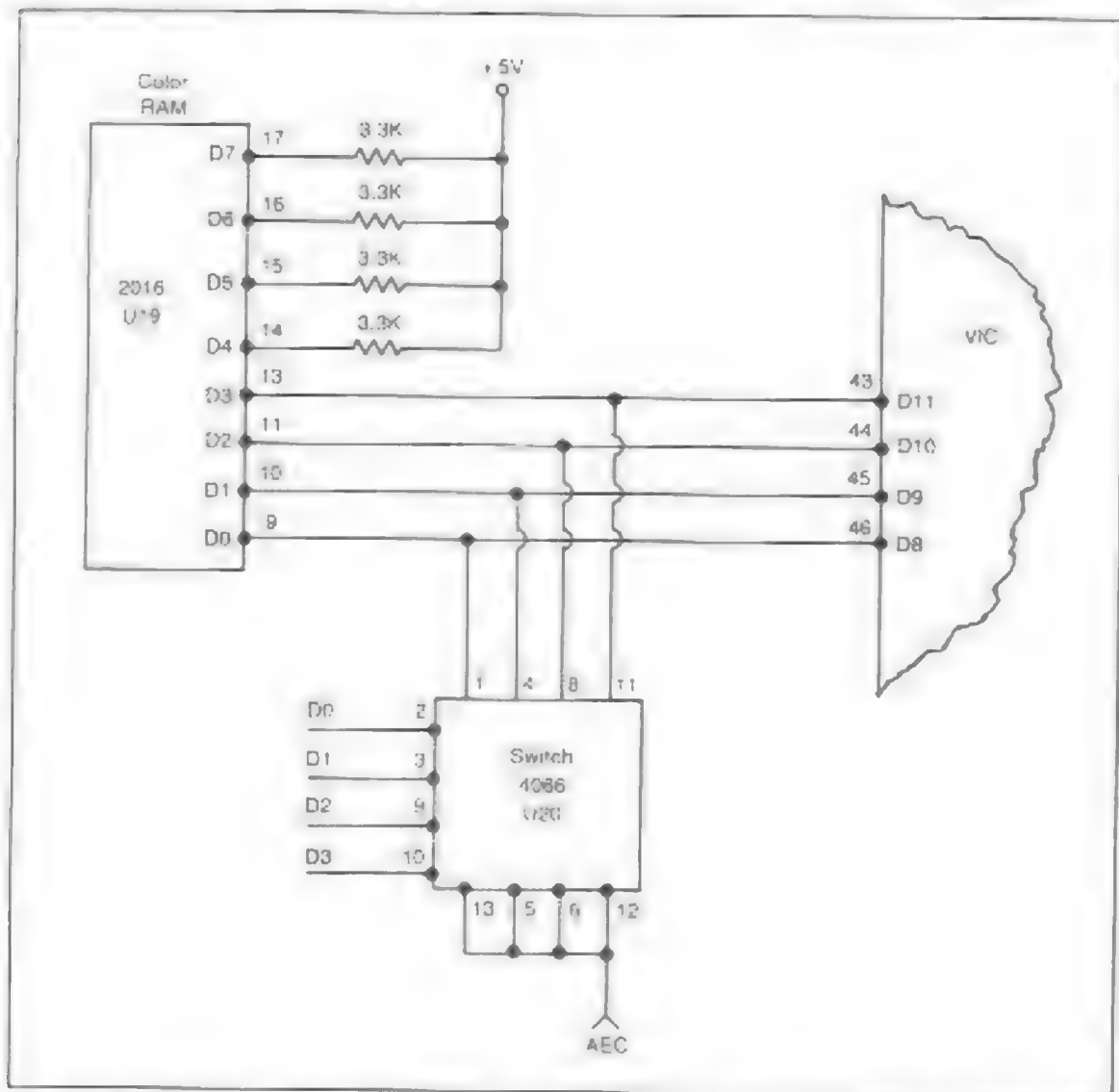


Fig. 18-7. When the Color RAM is accessed, it only uses four data lines to supply a choice of 16 different colors. The other data lines, D7-D4, are held high and are effectively out of the circuit.

switch chip. The 4066 has connections to the processor data bus lines, D3-D0, and connections to the color RAM chip, D3-D0.

When AEC is low, the 4066 switch is opened and the processor data bus disconnected. Remember, VIC must also be able to access the color RAM, wherein the codes for the color in the display reside. VIC pins 46-43 are its extended data bus, D8-D11. These four VIC pins also connect to the color RAM D0-D3 pins. When the processor data bus is switched out during AEC low, VIC is able to access the color RAM data nybbles.

The rest of the map, all with uncomplicated byte-sized registers, is connected to D7-D0 of the processor data bus. The bus is bidirectional. The 8502 is able to read and write on the bus. Every location is connected to the same data lines. Therefore, all the D7's, all over the map, are wired together, all the D6's are connected, and so on. Most of the time the locations are dormant. When a location is addressed, then that location, out of all of the 64K in a bank, is accessed. It is read from or written to. The rest of the locations are turned off and do nothing.

The data bus buffer in the 8502 originates its data bus. The buffers can receive data from the bus or transmit data to the bus. The instruction register is also connected inside the 8502, but it is an input only device. It can only receive instruction data—not transmit data to the bus.

The buffers and IR in the Z80 operate in the same way. However, as mentioned earlier, the Z80 can't interface directly with the 8502 designed bus lines. It is necessary to use U12, the 74LS373 latch, for the data entering the Z80. The latch connects directly to the D7-D0 system bus. It is also necessary to use U13, the 74LS244 buffer, to amplify the data leaving the Z80. The buffer chip also connects directly to the D7-D0 data bus.

Between the latch and buffer chips, and the D7-D0 pins on the Z80, is a small set of bus lines called ZD7-ZD0. This ZD bus connects to the latch, the buffer and the Z80. The wiring is direct. When AEC is high, and the Z80 is writing to the memory map, the data leaves the Z80, goes to the buffer, is amplified and then is placed on the system data

bus lines: D7-D0. When the Z80 wants to read a location, the data leaves the addressed location and enters the latch. Then at the proper time, designated by the system clock, the data is given to the Z80.

The data bus, once clear of the processors, splits off into many branches. An important branch goes to the 16 4164 RAM chips. The chips are separated into two sections, bank 0 and bank 1, each with eight chips. Each bank has chips numbered 0-7. One data line is assigned to each chip in the two banks. D0 makes two connections, one each to the two number 0 RAMs: D1 to the two RAM 1's, D2 to the two RAM 2's, and so on. That way, a byte can be stored with one bit in each chip of a bank. Pins 14 and 2 on each RAM chip are tied together and connect to one data line. One pin is for inputting a bit, while the other pin outputs a bit.

Other branches of the data bus connect eight lines to the ROMs, the CLUs, the Expansion Port, the MMU, VIC, the 8563 Video Controller, and SID. The data bus connections to VIC, the 8563 and SID will be discussed in their respective chapters again in more detail.

THE 80-COLUMN DISPLAY BUS

While VIC handles all the 40-column display chores, the 8563 Video Controller takes care of displaying 80 columns. Chapter 21 will discuss the 8563 activity in more detail. Meanwhile, the 8563 has its own private little buses that have little to do with the rest of the system buses. The 8563 is wired to two DRAMs, U23 and U25, a pair of 4416 chips, as shown in Fig. 18-8. These DRAMs have nothing to do with the 128K of system DRAM.

The 8563 originates two little buses, one for addressing U's 23 and 25, and the other to pass through data. The address bus is called the Display Address Bus and the data bus is called the Display Data Bus. Accordingly, their lines are DA0-DA7 for the address bus, and DD0-DD7 for the data bus.

The address bus, DA0-DA7, leaves the 8563 at pins 26-33. It connects to each 4416 at pins 14, 13, 12, 11, 8, 7, 6, 10. The 4416's are special 16K dynamic RAMs with four-bit registers. The data bus DD0-DD7 leaves the 8563 at pins 34-42 except for 37, which is V_{cc} .

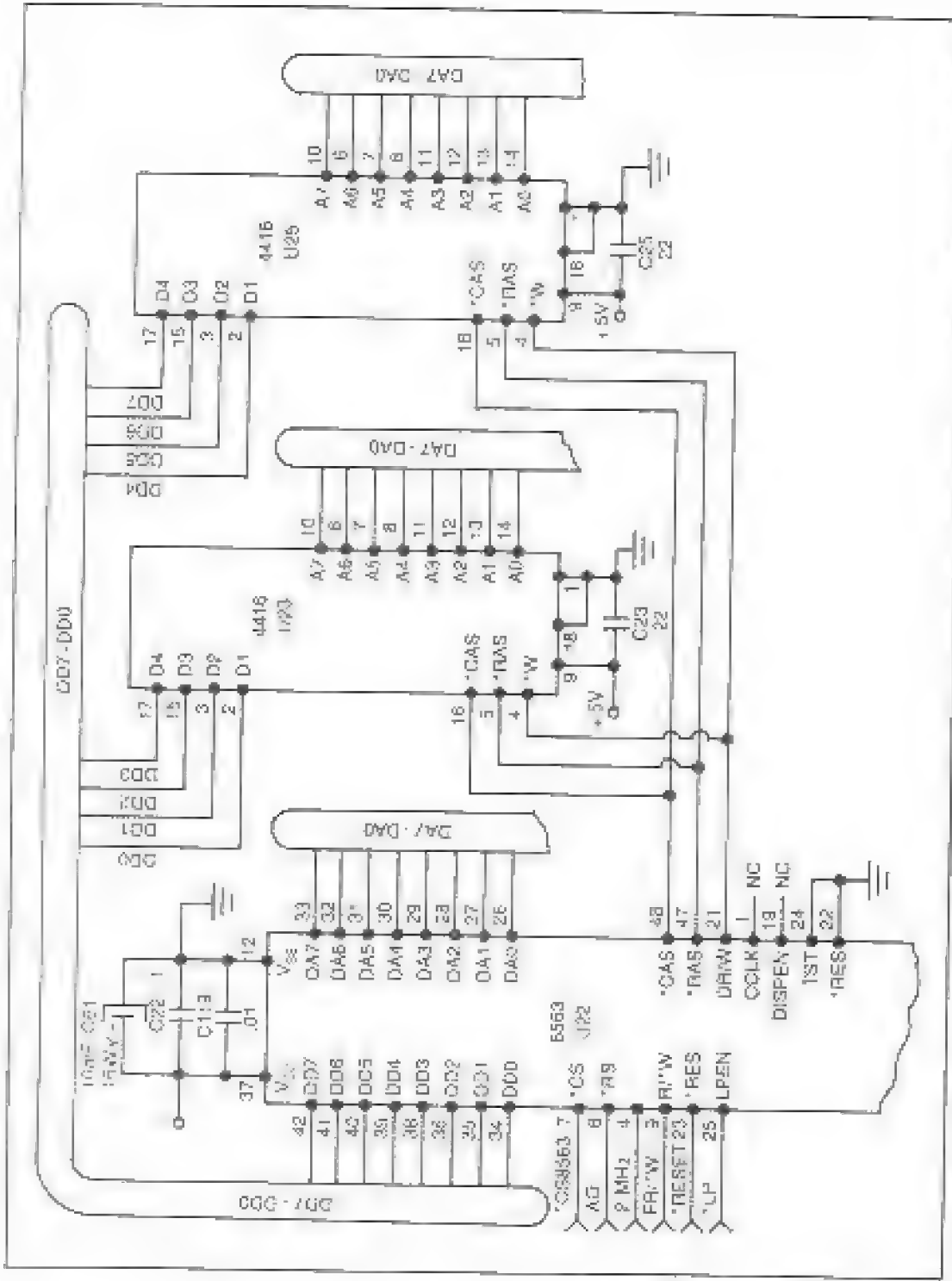


Fig. 10-8. The B563 Video Controller chip, U2, has its own private 119 address and data bus that has nothing much to do with the other address and data bus lines.

The data bus lines are then split in half. DD0-DD3 are connected to U23 and D14-DD7 to U25. Each DRAM supplies a nybble to the data bus which, together, are the byte that the 8563 wants to use. That way, the 8563 has a DRAM capacity of 16K bytes between the two 4416's. The 8563 operates with these DRAMs to produce the 80 column display.

The 8563 does the entire job for the two DRAMs. It supplies row and column strobes out of pins 47 and 48. It also provides their refresh signals.

THE CONTROL LINES

The many control lines coming and going from the 8502, Z80, VIC, the Video Controller and other chips are collectively referred to as the Control Bus. Actually, the lines are all individual with each one doing its own specific job. They are unlike the address bus and data bus systems in that each has many lines all doing the same job.

The various lines in the different chips are covered in the chapters on those chips. There are four lines though that connect to a number of chips and are worthwhile reviewing even though they have the chapter coverage. These lines are the R/*W, the *IRQ, the RESET line and the clock lines.

R/*W Line

The best known control line is the R/*W. It is an 8502 output that travels to all the major chips in the memory map. It connects all the R/*W pins on the concerned chips together. The entire line is normally held high by a 4.7K resistor connected to +5 volts. When high, the R/*W line is in a read mode. Note in the name that the R does not have an asterisk. Therefore when this line is high, the R (Read) is active. When the line is forced low by the 8502, then the W (Write), that does have an asterisk, is active. The R/*W line determines which way data is to travel on the data bus.

The line, as shown in Fig. 18-9, goes to the RAM chips, the CIAs, MMU, PLA, VIC, the 8563 Video Controller, SID and the Expansion Port. The R/*W signal originates in the 8502 instruction decoder. It connects internally to the data bus

buffers and is then output from pin 39. It is held high most of the time as the 8502 reads from the memory map. When it is time for the 8502 to write to the map residents, the instruction decoder forces the line low.

*RD and *WR

The Z80 originates separate read and write control lines, shown in Fig. 18-10. The *RD and *WR lines out of the Z80 do the same type of job as the single R/*W line out of the 8502. The *RD line out of pin 21 connects directly to *E: enable pin 1 of the latch U12 (74LS373). When the Z80 wants to read, *RD gets the latch to operate.

The Z80 *WR pin 22 is held high by a connection to +5 volts through a 3.3K resistor. *WR also connects to some gates and then the 74LS74 clock circuit flip-flop. This flip-flop then outputs to a NAND gate that, in turn, connects to an enable pin 1 of U13, the 74LS244 buffer. That way the *WR has control over the buffer that interfaces the Z80 data bus to the system data bus. When the Z80 wants to write, *WR gets the buffer working.

The *IRQ Line

The *IRQ line is an external input to the 8502 and the Z80. The *IRQ, as in Fig. 18-11, can originate from CIA1, VIC, or the Expansion Port. The line is held high by a connection to +5 volts through a 3.3K ohm resistor. When one of the origination points wants to interrupt the 8502 or Z80, whichever is in charge, it sends a low over the line.

A peripheral connected to CIA1 can send an interrupt through the I/O chip. VIC could possibly need an interrupt and would then generate a low. A cartridge in the Expansion Port could be the originator and send a low over the line.

Whatever the reason is, if the 8502 suddenly is input a low at pin 3, then it will go into its interrupt service routine as described in Chapter 12. Should the Z80 be in charge, the low will travel to pin 16, the Interrupt Request pin, and it will do what it must do for the interrupt.

The interrupt must be considered when you are conducting machine language programming. It per-

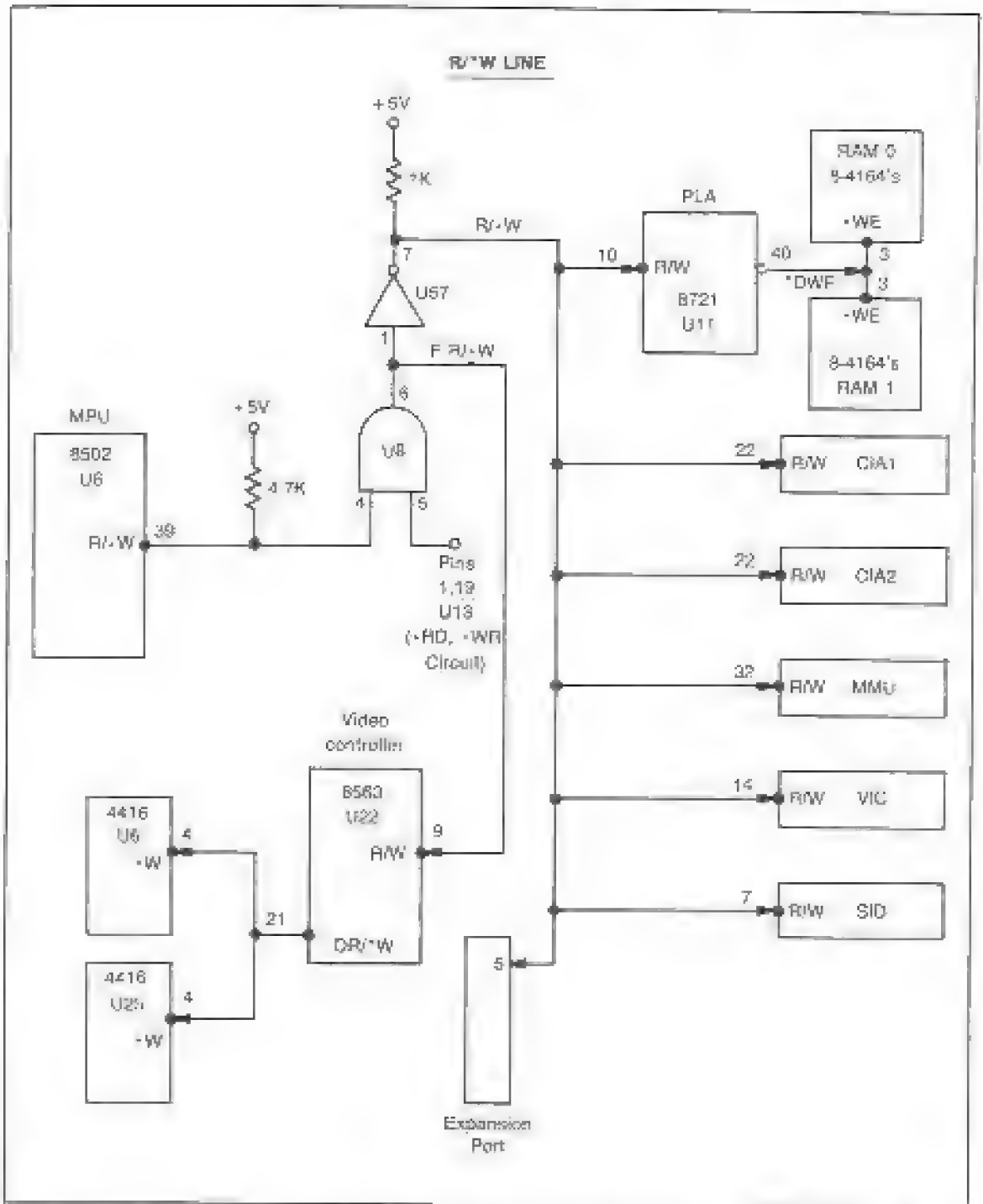


Fig. 18-9. The R/W line originates in the 8502 and connects to all the residents of the memory map except the ROMs.

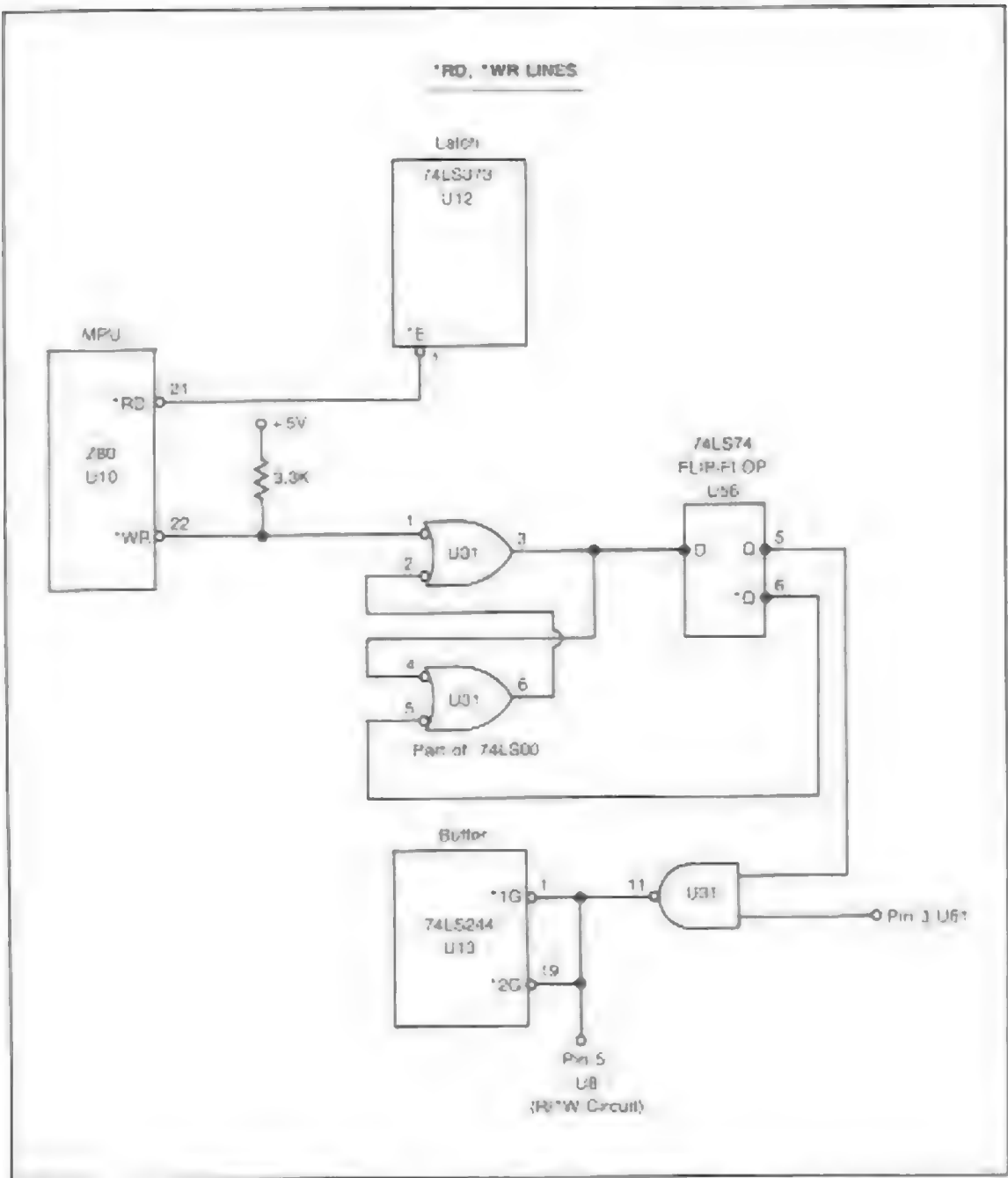


Fig 18-10. The ***RD** and ***WR** lines are the Z80 counterpart of the 8502 **R/W** line. Line ***RD**, the read line, connects to the U12 latch to conduct processor reads. Line ***WR**, the write line, connects to the U13 buffer and then joins with the system **R/W** line at U6.

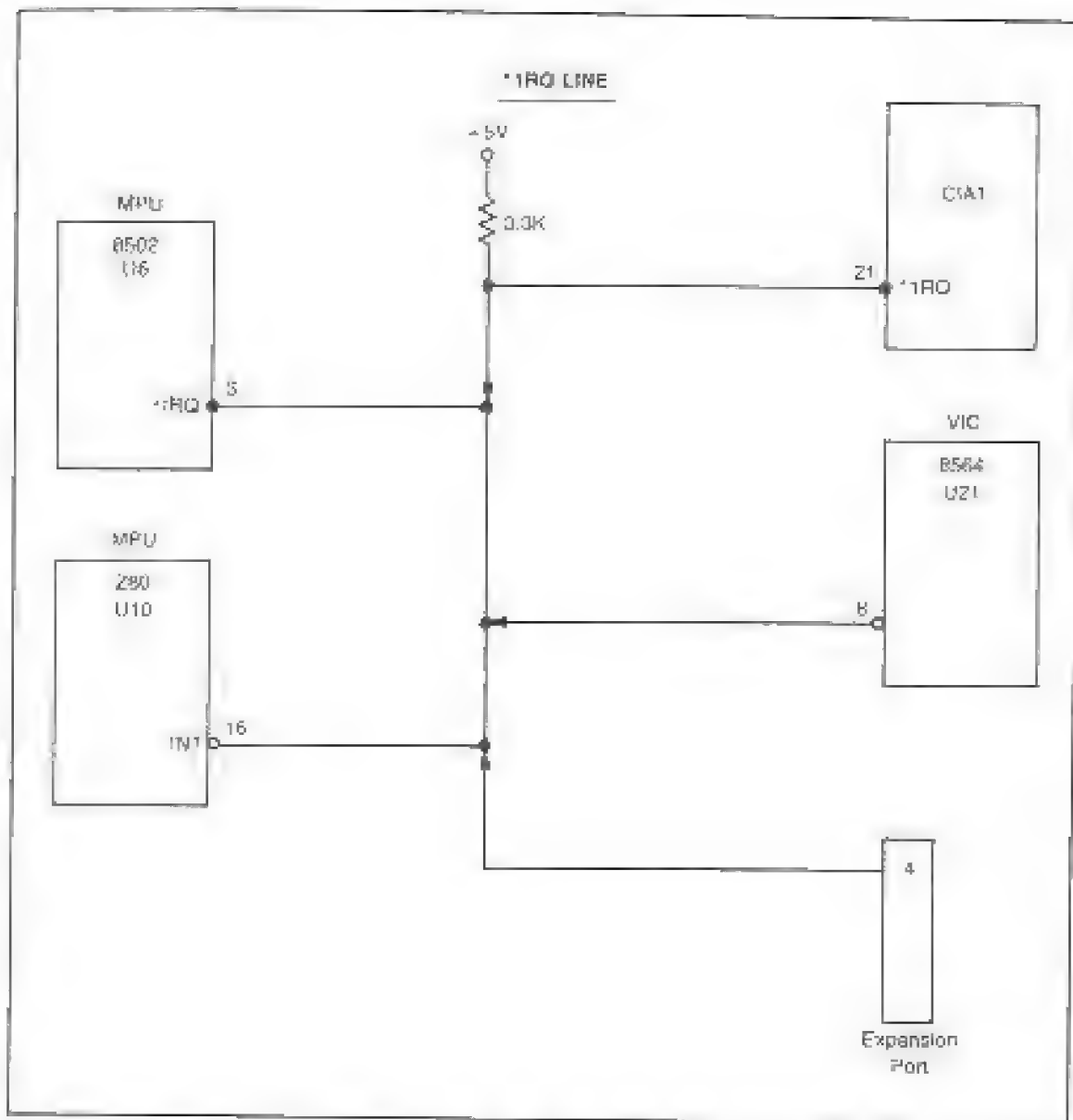


Fig. 18-11 The Interrupt request. *IRQ originates in CIA1, VIC or from some chip in the Expansion Port. It can interrupt both processors.

forms automatically if BASIC is being used. For troubleshooting, you need to know where the lines are on the printboard and to look for the high or low that should be present under your test conditions.

The RESET Line

The RESET line is used to get the computer underway when it is first turned on. The line originates at the 556, timer chip U27 (see Fig.

18-12). There are two timers in the chip. One is for the #NMI line covered earlier and the second, in Fig. 18-12, is for the RESET circuit.

When the C128 is first turned on, the +5 volts connected to the trigger pin 8 starts to build. As the +5 volts develops, the trigger is pulled, a high is generated and output from pin 9. The high is passed through a NOT gate and sent out over the line. The line is otherwise held high by a 1K resistor connected to +5 volts.

*RESET is attached directly to the 8502, Z80, MMU, 8563 Video Controller, the two CIAs, the User Port, the Serial Port, the Expansion Port and SID. When these RESET destinations receive the low, they do not operate. They are turned off. Then as the line returns to its held-high state, the processor goes into its RESET service routine. This initializes the C128 and prepares it for duty.

The C128 also has a RESET button. The C64 computer does not have a RESET button. The button is a switch connected to the trigger pin of the 556. If you press the button, the connected parties all turn off and lose their initialization. When you let up on the button the C128 starts all those circuits over again. The registers are reinitialized.

The Clocks

As discussed in Chapter 17, VIC generates all the system clocks after it operates with the crystal to produce one of the two master frequencies either for American or European use. The main clock derived from the chosen frequency is the 1 MHz. All of the I/O operations use the 1 MHz clock. In addition, the majority of the system bus lines move bits to the 1 MHz beat.

The clock leaves pin 18, exits pin 23 (confusingly called the 2 MHz) and is processed through a 68 ohm resistor and called D1MHz. The clocks are connected to the 8502, Z80, the CIAs, 8563, SID, MMU and the Expansion Port. The clocks do all the driving of the bits that travel from place to place in the computer. This keeps all operating parties in perfect sync with each other. The clock activity can be found at many test-point pins reading PULSE on the logic probe.

TESTING THE BUS LINES

Because the bus lines in any microcomputer cover a lot of printboard real estate, they have a lot of exposure to trouble. In the factory, when computers are made, the bus lines can be the source of a lot of problems. Most of the damage is created during the soldering operations. Sometimes the solder spreads all over and forms balls, threads, and other odd shapes. The solder can drape itself over the top and bottom of the board and create havoc. The solder can get between traces, from trace to ground and in all the tiny cracks and crevices the components and chips reside in.

In the early days of printboard manufacturing, this was an expensive problem, but newer techniques have fortunately eliminated the majority of these troubles. However, the situation still requires that troubleshooters check out the solder drip problem during routine checks. When trouble strikes, the bus lines are still prime suspects. Solder shorting still happens on occasion; open traces still occur; and the components in the bus lines could fail.

As you can see, the bus lines are the wiring of the computer logic. One of the first tests you could conduct is the continuity of the copper trace bus lines.

Continuity tests are, of course, tests of the resistance in ohms of the copper lines. The resistance of the copper to electric flow is so low that your ohmmeter would read near zero ohms. This means the line just tested is intact between the two probe touchdown points. On the other hand, if there is no continuity or the ohmmeter reads infinite resistance, then this indicates that the line under test has a break and will not pass electric current.

These tests are easy with an ohmmeter. Unfortunately the ohmmeter does not give a reliable test when there are integrated solid state circuits connected to the bus under test, and all the buses will have ICs connected.

It so happens that there is usually a 1.5 volt battery driving the ohmmeter. When you touch down with the two probes, the 1.5 volts is pushed through the line under test. Transistors turn on with voltages well under a volt. Silicon transistors turn on with the

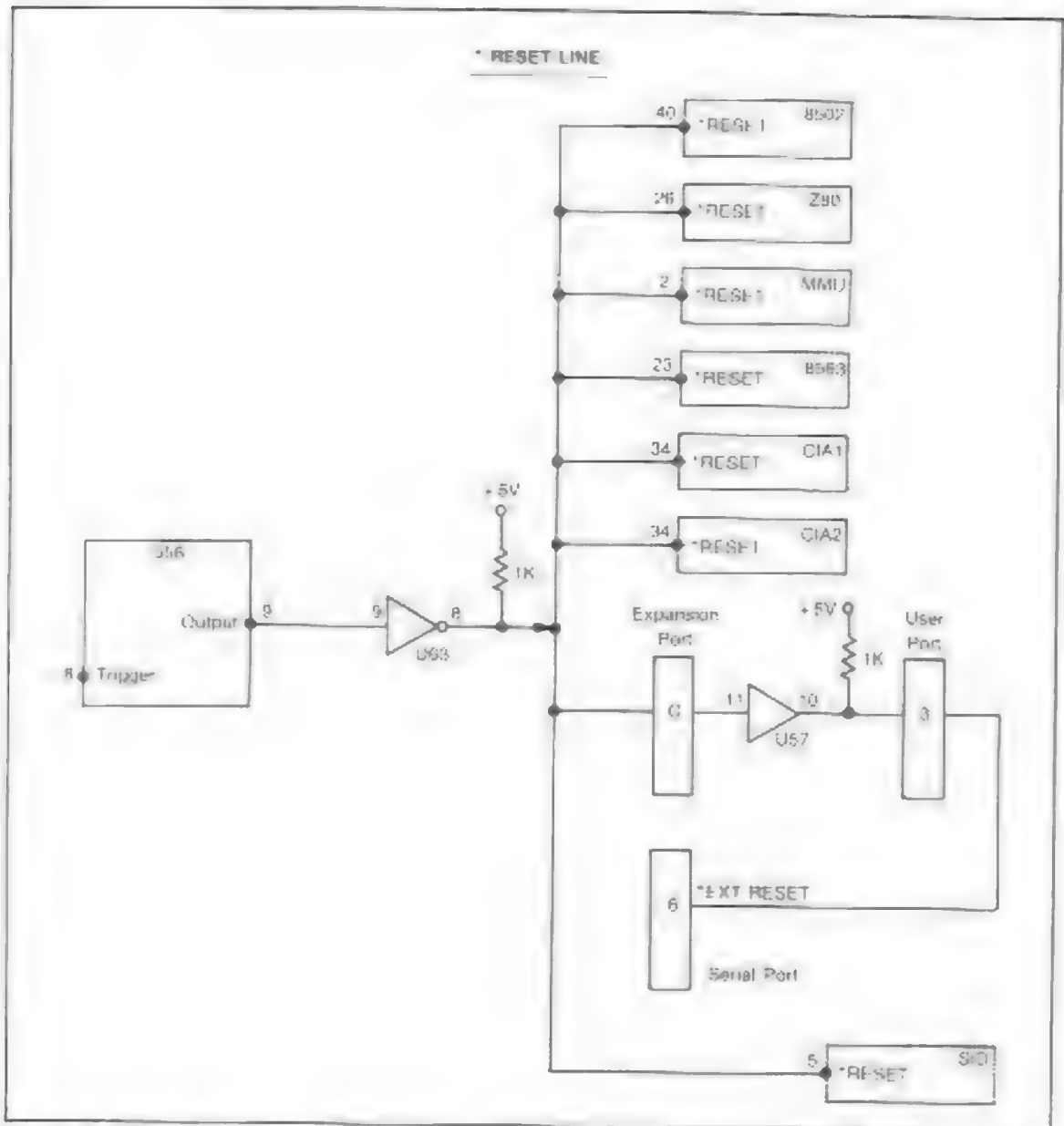


Fig. 18-12 When the -RESET signal is activated it turns off these chips and then starts them all over

application of about 0.6 volts. The 1.5 volts could get many transistors operating somewhat and produce false readings on the ohmmeter. Its best not to use the ordinary ohmmeter for continuity testing of bus lines.

If you do not have a special type ohmmeter to do these tests, then one can be made easily. It is called a *low voltage continuity tester*. The diagram for it is in Fig. 18-13. The parts are inexpensive and easily wired together. It will only push about 0.25

For digital circuits
low voltage continuity tester

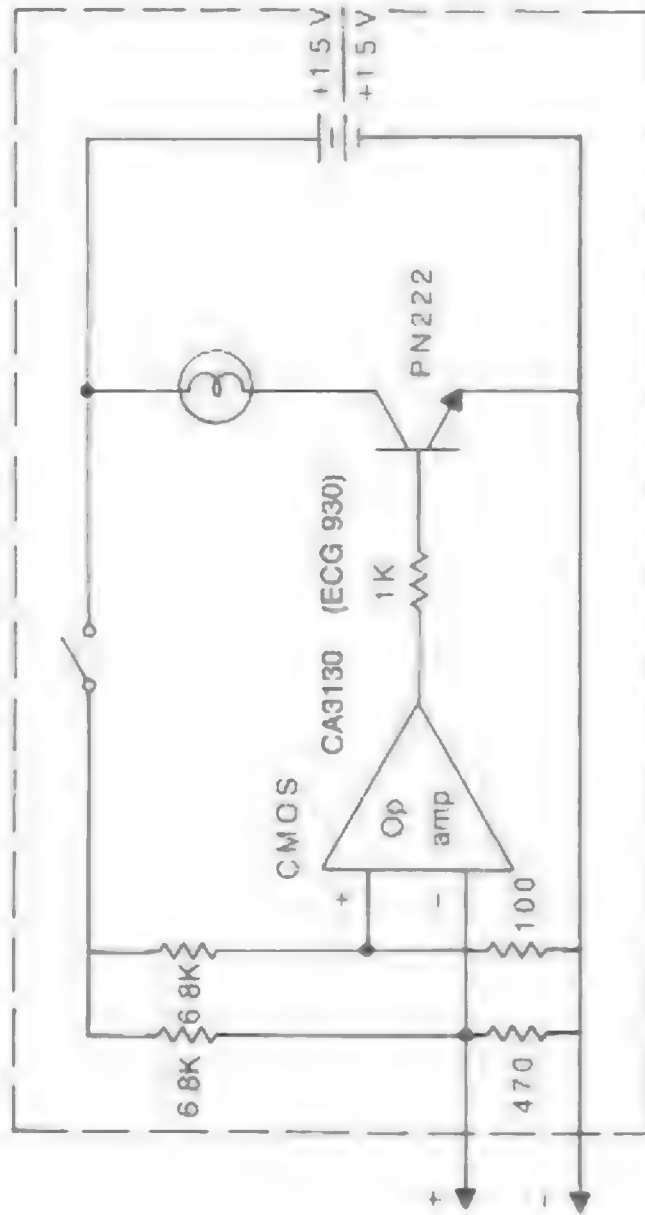


Fig. 19-13. It is risky using an ordinary ohmmeter in circuits with $0.01\ \mu\text{F}$. It is best to use a low voltage continuity tester. And, this easily assembled circuit, you can test for continuity and only push 25 volts through components instead of the usual 1.5 volts that an ohmmeter uses.

Table 18-1. Go-No-Go Continuity Chart.

Service chart		
SYMPTOM-CONSTANT PROGRAM ERRORS		
RESISTANCE TEST	GO	NO-GO
POINT TO POINT	ALL POINTS CONTINUOUS	OPEN TRACE
TRACE TO GROUND	ALL RESISTANCES FAIRLY HIGH AND ALIKE	A TRACE READS SHORT TO GROUND
TRACE TO TRACE	ALL POINTS ARE HIGH RESISTANCE	SHORT BETWEEN TRACES

volts through the sensitive circuits. This will probably not affect any nearby silicon junctions. The tester will read the two required resistance conditions. If the line under test is continuous, near zero ohms, then the bulb will light. Should the line be open, the bulb will not light.

Besides testing the continuity from test point to test point, a second type test is from a trace to chassis ground. If a trace is shorted to ground, and it is not supposed to be, then the test bulb will light. Should the trace be okay and not shorted to ground, the bulb will not light. Table 18-1 shows the go/no-go test results. With the tester, the bus lines can be checked out for opens or shorts very quickly.

Using the Tester

The continuity tester is one of the most used pieces of test equipment during troubleshooting. The first thing to make sure of is that the C128 is com-

pletely disconnected from power and peripherals. Pull the power supply ac plug out of the wall. Also disconnect the power supply unit from the side of the C128. You do not want any energized circuits during testing. This could light the tester's bulb and cause confusion.

The best place to view the circuit traces is from the top of the printboard. The bottom of the board is, for the most part, where the solder connections are. In the C128, a number of the chips are socketed. It is often useful to remove the socketed chips, with great care, to get a good resistance test. When the chips are out of the way, you are sure that the readings are of the board wiring and soldered components only.

The first tests that should be made are point-to-point continuities. Once the 16 address lines, the eight data lines, and other bus lines are cleared for opens then you can run tests for shorts. The method

is not as foolproof as finding an open trace, but it is worth the effort. Measure the resistance to ground of all the address and data lines one by one.

The results should all be the same. All the trace to ground readings should be a high resistance and not light the bulb under normal circumstances. If you use an ohmmeter, then there would be a high resistance at each trace to ground, and there could be a slight variation in resistance but still a very high resistance. Should you find one of the traces quite different than the other, then that could be a valid clue that the trouble is nearby.

Locate the circuit involved with the trace and with the aid of the Master Schematic. A reason for the difference in the resistance to ground is possible in some odd case, but I haven't encountered any in different models. Chances are, if one of the traces shows a dramatically lower resistance than its companions, then you have pinpointed a short that is causing the trouble.

Once the resistance to ground of each trace is tested, check the resistance between traces. Here again there should be a high resistance between all traces. Should you find a low resistance or a short indication, this is a clue. Probably some sort of short, like a sliver of solder, is between the two traces that has developed the low resistance.

Bus Line Writing and Reading Tests

The continuity tests can be run under any circumstances. Just be sure that the C128 is completely disconnected from all power and peripherals. The continuity tests are run on a dead computer. The only energy applied is the tiny amount of current from the tester's battery. They are the static tests.

Dynamic tests are run with the computer energized. When the C128 is operating but is not addressing properly, or is delivering incorrect data, you could use the writing and reading tests. These are the BASIC PEEK and POKE tests or the Monitor's MEMORY and FILL tests.

Each bus line can be tested individually. For example, if you want to test a particular address line, all you have to do is address a key location, write some data to the register and then read the contents

and see if the data did indeed arrive and get stored in the location. A key location is a register that needs the address line being tested in the address bits. That way, the processor sends a high over the suspect address line to open the register. If the register responds, then that address line is okay. Should the register not be reached, it could be because the suspect line is open or shorted and cannot carry the address bit.

To review, 16 address lines leave the processor. Each line is capable of transporting a high or low. Actually, only highs are sent. When a line is not carrying a high it is considered as carrying a low. A line, therefore, is deemed okay if you can send a high successfully from the processor to the memory location addressed by the high.

Table 18-2 shows the addresses in decimal and hex that send a high over one address line while all the rest of the address lines are carrying lows. Therefore, if you output an address on the bus that has all lows (except for the copper trace you want to test, which will carry a high), and the address is accessed, then that trace is doing its job okay. If the address cannot be accessed, then the trace being tested is not transporting the high and could be the seat of the trouble.

The addresses in Table 18-2 test one address line each. The line carrying the high is the one being tested. For the address bus test, it does not matter what combination of bits you send to be stored and then read. The addressing exercise is the test, not the data sent.

To get the address into the machine, all you need to do is use the decimal code for the desired binary bit address and, in BASIC, POKE the address with any byte of data. If you want to use the Monitor, use the hex code for the binary address and FILL a couple of sequential addresses including the line to be tested.

The addresses shown are all in RAM. If for some reason you are testing an address that lands on a ROM then you can dispense with the POKE or the FILL. Just PEEK or MEMORY the address and read the contents. If the contents make sense, then the line carrying the address high is okay.

Table 18-2. Address Bus Test Chart.

TEST Poke (Copper Address Line)	Binary Address	Decimal Address	TEST Peek
A0	0000 0000 0000 0001	1	RAM
A1	0000 0000 0000 0100	2	RAM
A2	0000 0000 0000 0100	4	RAM
A3	0000 0000 0000 1000	8	RAM
A4	0000 0000 0001 0000	16	RAM
A5	0000 0000 0010 0000	32	RAM
A6	0000 0000 0100 0000	64	RAM
A7	0000 0000 1000 0000	128	RAM
A8	0000 0001 0000 0000	256	RAM
A9	0000 0010 0000 0000	512	RAM
A10	0000 0100 0000 0000	1024	RAM
A11	0000 1000 0000 0000	2048	RAM
A12	0001 0000 0000 0000	4096	RAM
A13	0010 0000 0000 0000	8192	RAM
A14	0100 0000 0000 0000	16384	RAM
A15	1000 0000 0000 0000	32768	RAM

Table 18-2 shows the binary addresses, the decimal code for the binary, the hex code for the binary and the type of register I found in the C128 for the test examples. In newer C128 models, there could possibly be different chips at those addresses. Check the memory map of your machine.

The data bus can be tested in a similar manner. For the data bus test, however, you need to produce eight data bytes that have all lows except for the data line tested. That line will have a high.

Table 18-3 shows the eight bytes of data that will test each data bus line. All that is needed is to POKE or FILL the eight bytes into any group of eight RAM addresses. Next, PEEK or MEMORY the eight locations. The display will show the results of the tests. If all the written bytes are correctly installed in their assigned locations, then the data bus lines are passing the data from the processor to the locations okay. Should one or more of the locations not contain the correct contents, then the data bus line or lines that were to carry the high to that location could be shorted or open. If there are any latches, buffers, gates, multiplexers or any chip large or small in the indicated line, that are supposed

Table 18-3. Data Bus Test Chart.

Copper Data Line	Binary Data	Decimal Data
D0	0000 0001	1
D1	0000 0010	2
D2	0000 0100	4
D3	0000 1000	8
D4	0001 0000	16
D5	0010 0000	32
D6	0100 0000	64
D7	1000 0000	128

to pass the bits, then they are also suspect. For instance, if the binary byte 01000000 did not arrive at its addressed register, then data line D6 could have trouble.

Control Bus Tests

The lines called the Control Bus can be analyzed in an indirect way when the writing and reading tests

are performed in BASIC or in the Monitor. For instance, PEEK is a function that sets off a BASIC ROM routine that reads an address. POKE is a command that sets off a BASIC ROM routine that writes data to an address. If you can successfully write and read to addresses, then the R/W line from the 8502 is working okay.

Using the same reasoning, if you can enter CP/M then the RD and WR lines in the Z80 are okay. The exercising of the usual procedure shows that the control lines involved are working. Should one of the routine procedures fail to perform, then you can puzzle out what lines are involved in the process and consider them suspects till they can be tested with the vom or logic probe and be identified as okay or not.

Because the control lines are all individuals, the best tests for the control lines is to find their origination and destination places and test along with the logic probe. Start at the origination point and proceed step by step to the destination. You should be able to follow the line under test and reason out what logic state should be present.

For example, suppose you want to trace out the NMI. The symptom could be: when you press the RESTORE key nothing happens. In Fig. 12-19 it can be seen that the RSTK line coming from the RESTORE key is held high by R9, a 10K resistor, to +5 volts. If you touch down there with the logic probe then you will read a HIGH. In series with the

line is U16, a trigger, and U37, a NOT gate. The trigger has a NOT circle. The high goes low but then goes high again after the NOT gate. The line should read a HIGH then after the NOT gate.

If these readings are okay then the next stop is the input of the 556, U27, pin 6. According to the U27 Test Point Chart it should be a high. If it is then test pin 5—it should be a low. From there you encounter a section of U29, a NOT gate. This inverts the low to a high. That part of the line is also held high by the 1K resistor connected to +5 volts. From there, NMI goes to pin 21 of CIA2, the Expansion Port, and pin 4 of the 8502.

There should be highs on all three of these test points. If you find a wrong logic state somewhere along the test path, then you have just passed over the troublemaker. Test the parts and check the connections there and you will probably locate the bad part or connection.

When you are signal tracing in this way, you have the aid of the Master Schematic and the Test Point Charts. When the pathway connects to a chip with a U number, double-check what logic state should be present on that pin on the Test Point Chart. Sometimes the states that look like they should be present have a circuit quirk and the opposite state is actually present. The Test Point Charts will keep you straight. When troubleshooting, a clue that is not a clue could easily appear. Don't let one fool you.

19. Complex Interface Adapters

The Complex Interface Adapter is packaged as a 40-pin chip. It has the job of interfacing the digital circuits with all of the external devices that the computer has to deal with. These are inputs such as the keyboard and joysticks and outputs like the serial bus and user port.

Inside the digital part of the computer, the 8502, RAM, ROM, etc. all operate at matching speed, frequencies, voltages, and currents. The 8502 can interface with the rest of the digital circuits by simply attaching buffered pathways and begin operations. Unfortunately, the various peripheral devices do not usually have the same characteristics as RAM and ROM. The 8502 cannot just hook into the external devices. That is where the CIAs come in. The CIA has addresses and an 8-bit data bus connection on the computer side. The address lines and the data bus connect to the processor just as the memory chips do.

On the outside of the CIA, there are two byte size buses that interface with peripheral devices. Inside the CIA are registers that are built to receive the bytes from the digital circuits, take care of an

mismatching, and smooth the way for the streams of data to be output to the peripheral. The two outside buses also receive data from the peripherals and forward the data through the CIA to the internal data bus.

Besides these parallel I/O registers, the CIAs have a serial I/O register. This is a shift register that receives parallel data from the internal data bus and converts it to a special output. The output leaves through a single pin. The serial register can also handle a serial input and convert it to parallel.

In addition to all that I/O work, the CIA is a dandy little timing device. It has registers that act as interval timers and more registers that provide an hours-minutes-seconds-tenths of seconds real time clock.

ADDRESSING AND CONTROLLING

The CIA has four pins connected to the address bus as shown in Fig. 19-1. They are pins 35-38. They are register selects RS3, RS2, RS1 and RS0. They tie into address lines A0-A3. These address

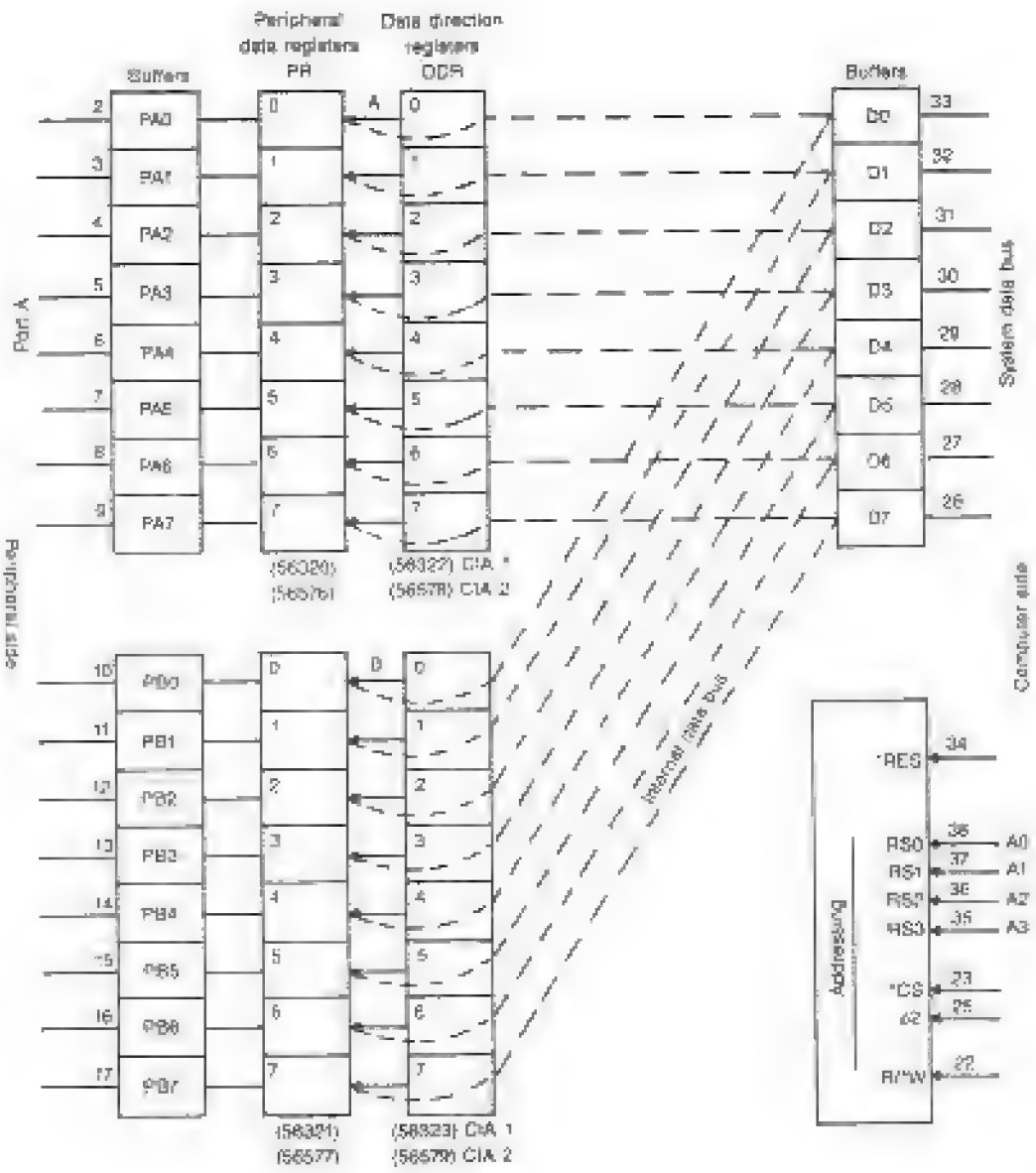


Fig. 19-1 The register selects of the CIA are connected to A3-A0. The chip select is made by the PLA and the 74LS09 decoder chip. However the chip select will not be enabled unless a ϕ_2 clock signal also arrives at the same time. Between the address bits that turn on the chip select and the address bits that choose a register, one of the 16 CIA registers can connect to the data bus. Four of the registers are shown here.

lines choose among the registers in the CIA. There are four lines that can select from 16 registers. The 16 registers in the CIA have decimal addresses, 56320-56335 for CIA1 and 56576-56591 for CIA2.

In addition to the internal addressing of the CIA, there is a chip select at pin 23: *CS. A low here will aid in selecting the chip. The signal for the chip select comes from the decoded signals out of the PLA and its accompanying chips (Master Schematic). The chip won't actually be selected unless a clock signal at pin 25 is simultaneously high. This is a timing reference so the CIA can be in sync with the pulsing of the data bus.

When the *CS is low and the clock is high at the same time, then the CIA will go to work with the R/*W line and the addressing bits. If R/*W is high, the 8502 can read the CIA. If R/*W on pin 22 is low, the 8502 is able to write to the CIA.

The *RES, at pin 34 is usually held high and is not active once the computer is operating. If the need be, and the *RES is forced low, all the internal registers in the CIA will be reset. The I/O pins are all set as inputs. The I/O registers are reset at zero. The timer control registers are made zero and the timer latches are filled with lights. All the other registers are cleared to zero.

INTERNAL DATA TRANSFER

The CIA is connected at pins 26-33 to the data bus lines D7-D0. These pins are held in a three-state condition until *CS is low and $\phi 2$ pulses high at the same time. With the R/*W line also high, the data from a peripheral will be pushed out onto the data bus and can be read by the 8502.

The CIA has an interrupt request line at pin 21 called *IRQ. It is normally held high by a pullup resistor to +5 volts and is inactive while high. This lets the line have a number of interrupts connected together. If one of the interrupts forces the line low the interrupt acts as described later in the chapter in the ICR discussion.

TIMING

The CIA responds to the addressing by the 8502 as the clock cycle goes from low to high and then back to low. This takes about 1000 ns. During that

time, the *CS pin is forced low, the R/*W line is made high for a read or low for a write, the RS pins receive a register address, and the data bus either inputs data or receives data. A timing diagram appears in Fig. 19-2.

The output pins in turn place data into the CIA for a read operation or send data outward if a write is taking place. All of this is shown in the timing diagram. The diagram is confusing at first glance, but taken step-by-step it can be comprehended.

The $\phi 2$ clock cycle controls the timing of the data transfer. The cycle time is, as mentioned, 1000 ns. This coincides with the accessing by the 8502. The $\phi 2$ signal has a high pulse that lasts 440 ns. Its low runs 420 ns. The rise and fall times are 25 ns each.

Write Timing

When the 8502 writes to a CIA, the timing of each signal is measured. The write operation begins by the 8502 addressing and activating the CIA. The processor outputs an address over the address bus. Bus lines A15-A12 enter the PLA and cause the PLA to output *IOCS to pin 5 of the 74LS238 decoder.

Meanwhile A11, A9, A8 enter pins 3, 2, 1. The chip then can output the CIA select signals, *CIA1 and CIA2, as in Fig. 19-3. These signals go to *CS of the two CIAs. The signals will enable *CS when low.

As the write cycle begins, $\phi 2$ goes high. For the first 58 ns, the addressing goes through an *address setup time*. The address bits are making their way through the PLA and decoder. Then the signal arrives at pin 23 of the CIA. The signal is a low. It turns on *CS. The low stays there for the remainder of the time $\phi 2$ stays high. That is 280 ns.

$\phi 2$ then falls to a low. As it falls, the low on *CS goes through a quick 10 ns *address hold time* and then rises to a high.

While the chip selection is going on, the registers that are to receive the data are addressed. Address lines A3-A0 open up one of the 16 registers in the CIA by putting address bits into pins 35-38, RS3-RS0.

Also during the $\phi 2$ high, the 8502 sends the R/*W signal to pin 22 of the CIA. The R/*W signal

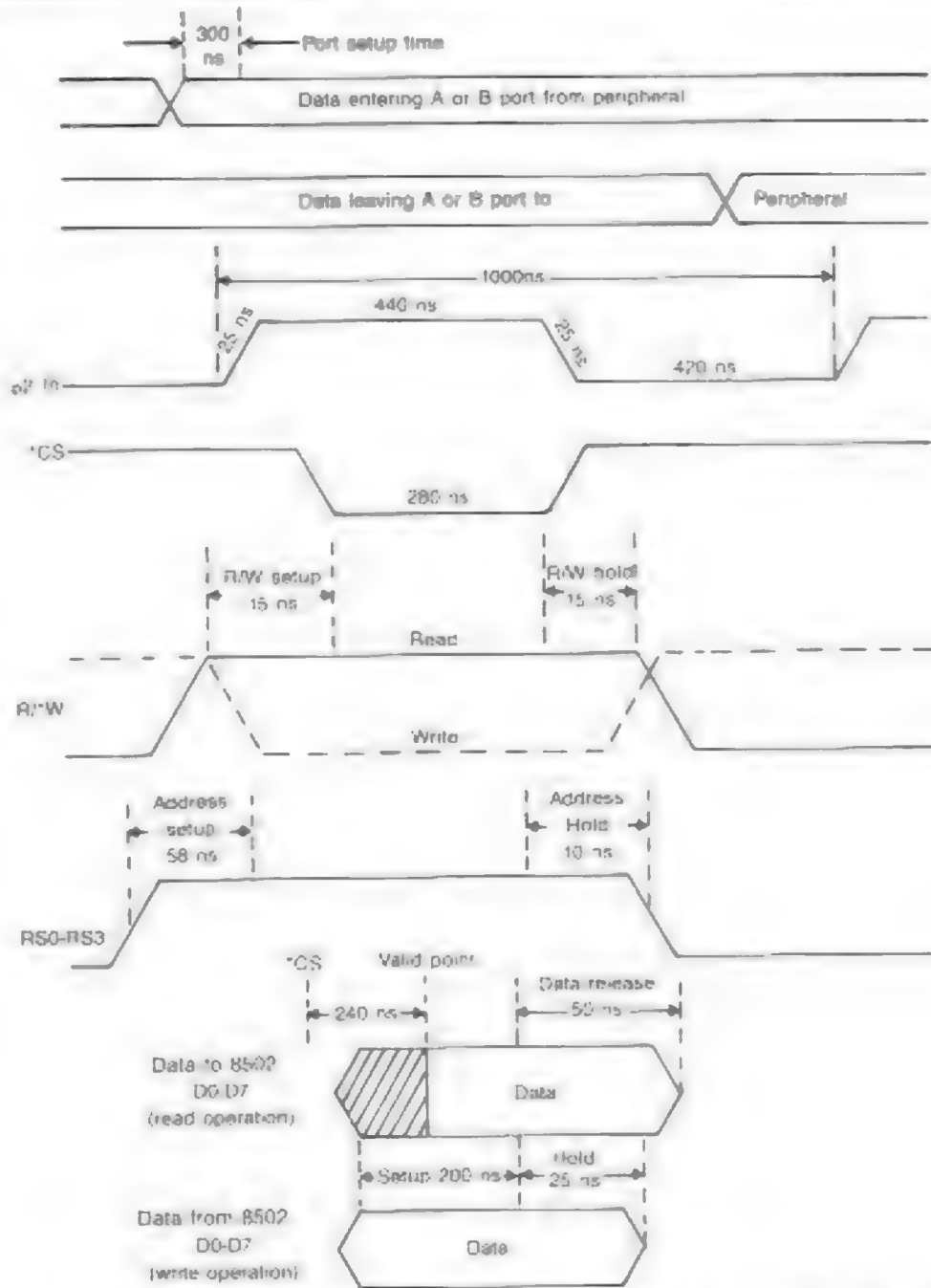


Fig. 19-2. The two top waveshapes are data entering or leaving one of the ports. $\phi 2$ must be high when that happens. \overline{CS} must be low. A read or write signal indicates which way the data must flow. The register select bits select one of the 16 locations. Valid data can then be fetched as $\phi 2$ falls.

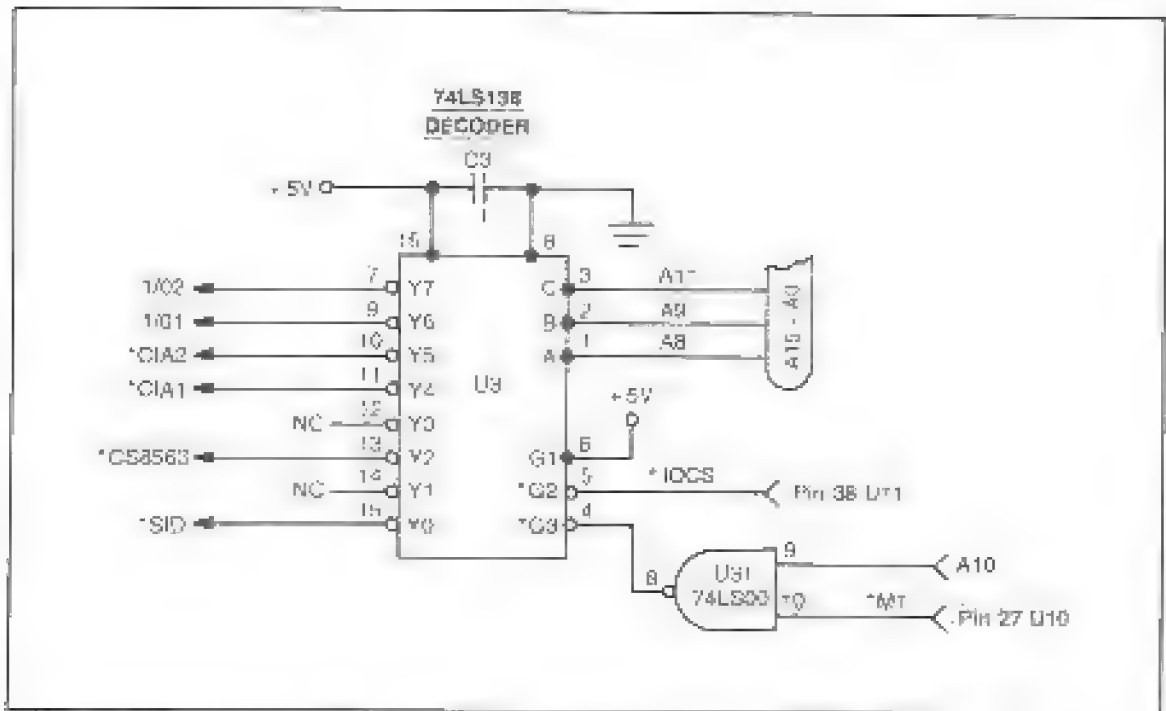


Fig. 1B-3 The 74LS138 decoder performs the CIA1 and CIA2 chip selects with signals *CIA¹ and *CIA²

starts off high but then falls as the write operation begins. The R/*W signal goes through a 15 ns R/W setup time as soon as it goes low. It stays low all during the remainder of the $\phi 2$ high. Then it has a 15 ns R/W hold time.

Once the chip is selected, the desired register addressed, and the R/*W line made low, the data can be written to the CIA. The data leaves the processor, goes out on the data bus, and arrives at the data bus pins of the CIA. It arrives about half-way into the high of $\phi 2$. The data must have a data setup time of at least 200 ns during the $\phi 2$ high. Then as $\phi 2$ falls the data is strobed into the CIA. The data then must remain valid for another 25 ns, the data hold time.

Meanwhile, all during the operation, the data output pins are waiting for the data to be received by the CIA and pass through it and emerge to the peripheral. The data output is able to remain in waiting for 960 ns of the 1000 ns total cycle time. This is plenty of time for the data to get through and out as the 8502 writes to a peripheral device.

Read Timing

When the processor wants to read from the CIA, it goes through a similar procedure. The main change is that the data will be traveling from the CIA to the 8502 rather than the other way around. During the read cycle, there are a lot of identical movements of signals and timings. The *CS low remains 280 ns during the $\phi 2$ high in the same way. The address setup and hold times are identical. The R/*W setup and hold times are also the same. However for the read operation, the R/*W goes high instead of low.

The read operation begins slightly before the $\phi 2$ cycle. First, the port pins must be open and available to receive data from the peripheral. This is called the port setup time. The ports must be setup for 300 ns. After the 300 ns stabilizing time the $\phi 2$ clock will then go high. The data enters the CIA through the ports and is ready to be read by the processor.

The $\phi 2$ clock at that point in time goes high. The CIA can then be selected, and the desired reg-

ister can be addressed. As \overline{CS} is forced low, a 240 ns time period begins. The data is let loose onto the data bus but will not be valid till that time elapses and the data settles in place. Once the 240 ns elapses, the data is valid and can be latched into the 8502. This happens as $\phi 2$ falls to a low. The data then has a short hold time to ensure stability. The hold time is called *data release time* and lasts for 50 ns.

As the processor accesses the CIA, it must do so while \overline{CS} is low and the register is addressed by RS0-RS3 signal bits. $\phi 2$ must be high, and the data must be valid. If any of these requirements are not met, the data will not travel from the peripheral device to the CIA and over the data bus into the 8502.

I/O PORTS

The \overline{CS} pin, $\phi 2$, R/*W, RS0-RS3, and I07-I00 pins all work on the computer side of the CIA and communicate directly with the 8502. On the peripheral side of the CIA, there are the pins that keep in touch with the external devices. There are two 8-bit I/O ports and a number of other pins. Let's examine the ports first.

It was mentioned that there are 16 addressable registers in each CIA. The registers are addressed internally through bits entering RS3-RS0. The first four registers in binary are LLLL, LLLH, LLHL and LLHH. These four addresses are to the four 8-bit registers that operate the ports. There are two ports, A and B. Each port has a Peripheral Data Register, PR and a Data Direction Register, DDR. Figure 19-1 shows which addresses contact what location.

The PR registers, through buffers, are connected to the port pins. PRA connects to pins 2-9, PA0-PA7, and PRB to pins 10-17, PB0-PB7. The DDR registers do not have external pins on the chip. They are connected internally to the PR registers, bit by bit. For instance, DDRA bit 7 connects to the PRA bit 7.

The DDR register bits go one way to their PR register bits. Bits do not travel the other way from the PR to the DDR. The DDR controls the PR. The DDR bits make the decision of whether a PR bit is

going to be an input bit or an output bit. If a bit in the DDR is set to a high then the corresponding bit in the PR becomes an output. Should a bit in the DDR be reset to a low then the same bit in the PR turns into an input.

Once the DDRs are programmed and turn the PR bits into the desired input or output mode, they are not used again unless a PR bit must have its status changed. There are internal circuits that set up the input or output mode of the PRs. During a read operation, the PRs are inputs and will latch the incoming data from the peripheral. For a write operation, the PRs are outputs and will conduct the data right on out to the peripheral.

*PC AND *FLAG PINS

The CIAs are able to transfer data with handshaking. *Handshaking* is a fairly sure way of receiving data from a peripheral or sending data to the peripheral. Handshaking is a programming chore, but during servicing, there could be times when you need to understand what is happening at the ports as the handshaking takes place. Pin 18, *PC, and pin 24, *FLAG, are deeply involved with the handshaking.

*FLAG is an input pin that connects to the internal data bus of the CIA and, therefore, to the inputs of both the A and B port circuits. *PC is an output pin that exits only from the B port through a special *PC buffer stage. *PC will output a low for one cycle after any read or write of the B port. Refer to Fig. 19-4. During handshaking, this automatic low can be used to talk to another device. The signal could be code saying, "data is ready" or "data has been received."

The input to *FLAG is also useful during the handshaking operation. If a low is sent to *FLAG, the signal sets an internal interrupt bit in the CIA. Setting the *FLAG bit can be used as a code signifying the readiness or reception of data too. As mentioned, handshaking is a programming job. If you want more details, check out a programming book.

TIMERS

There are two 16-bit interval timer registers in the CIA. They need four addresses since they are

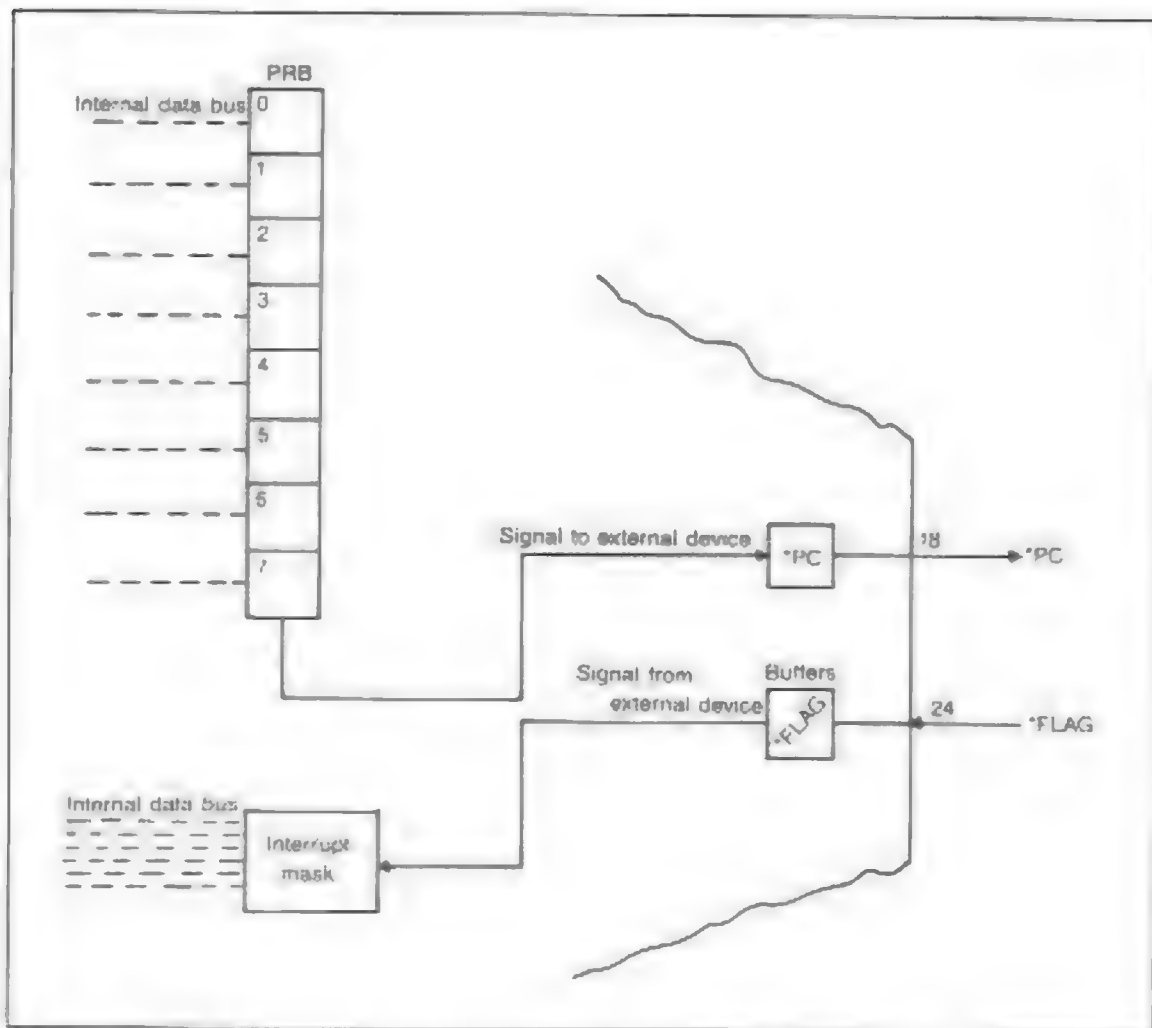


Fig. 19-4 *PC and *FLAG of the CIA are an output and an input that is used during handshaking

16 bits each, as Fig. 19-5 shows. Timer A has an 8-bit low register and an 8-bit high register. They have addresses of LHL and LHH. Timer B has a similar arrangement. Addresses LHL and LHH contacts the low and high bits of the register.

The timers have a number of modes. They are very useful in lots of applications. They can generate long time delays in a program, handle different width pulses, pulse trains, and waveform frequency changes. They can count pulses, measure frequencies and other characteristics of pulses that are injected into pin 40, CNT.

Each timer is controlled by another register in the CIA. Timer A is controlled by an 8-bit register at address HHL. Timer B is controlled by the register at HHH. The control registers run the timers. The two timers are similar in operation but not identical.

The timers need two addresses because they use 16-bit registers attached to the 8-bit data bus. In order to read or write to a timer, the 8502 first addresses the low register and accesses it. Then it addresses the high register and accesses it. Actually each timer consists of two 16-bit registers. One

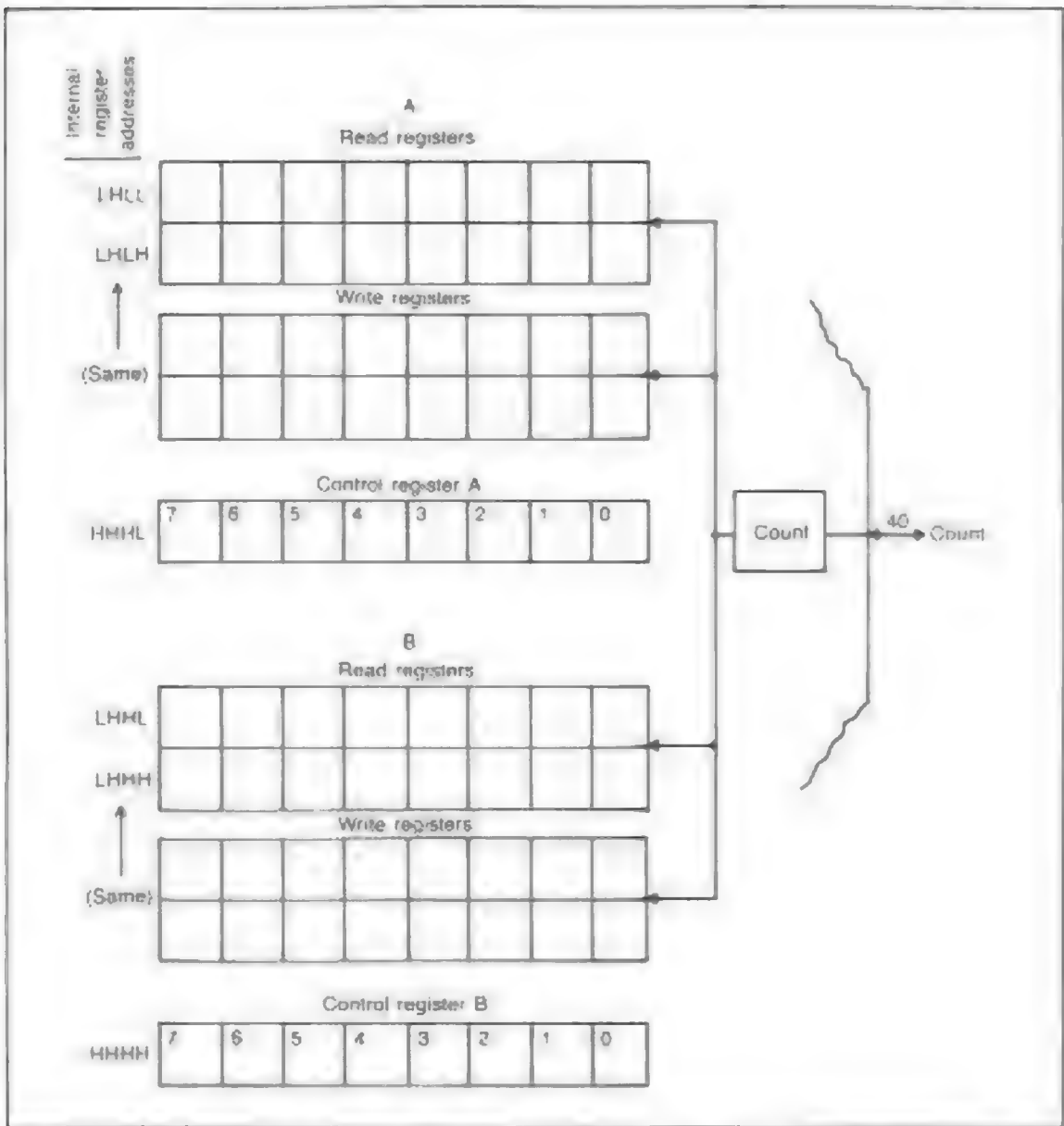


Fig 19-5. The timer registers can count pulses, measure frequencies, generate long time delays, and other things. They receive their data at pin 40, COUNT

is contacted during a write and the other when the processor does a read

The register that is written to is a latch. The register that is read is the timer counter. Anything written to a 16-bit timer location becomes latched.

When the same address is read the processor receives the current contents of the timer counter. Here again this action is important to the programmer. For servicing it is only of value to have an idea of what these registers do.

REAL TIME CLOCK

There are four registers in the CIA that act as a real time clock. They are shown in Fig. 19-6. During actual programming applications, a real time clock is often needed. This one has four 8-bit registers to handle the timekeeping. Address HLLL counts

10ths of seconds. HLLH takes care of full seconds. HLHL is the minute calculator and HLHH is the AM/PM hours register.

With these registers, the CIA can do 24-hour timing with a resolution of 10ths of a second. The four registers are an electric clock. Like any elec-

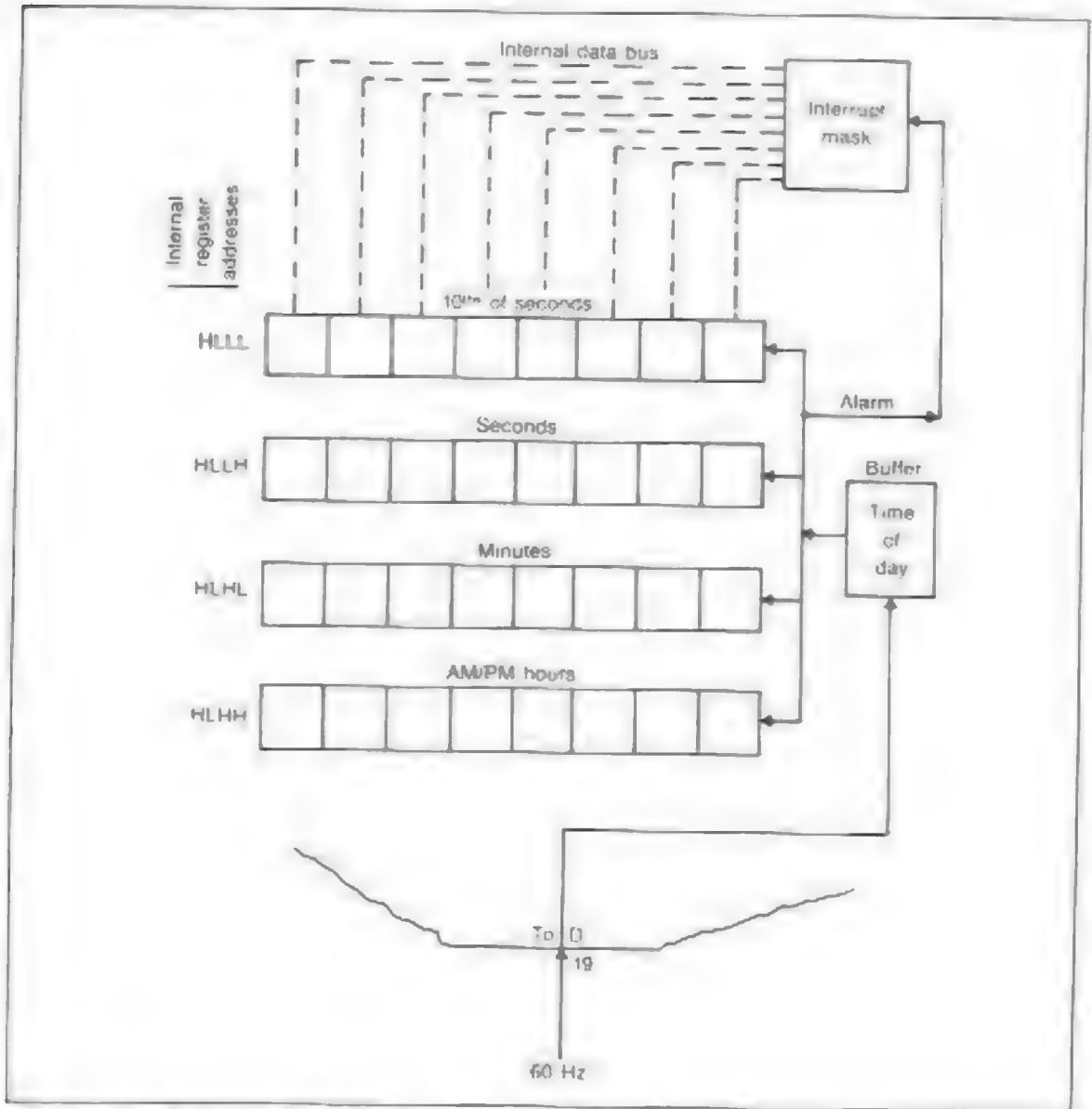


Fig. 19-6. Each CIA contains four registers that act as a real time clock. They count as fast as tenths of seconds and are controlled by a 60 Hz pulse into pin 19 from the power supply.

tric clock it is plugged into the electric company and uses the 60 Hz voltage to time the clock. That is, 60 Hz in this country, other nations could use 50 Hz. The 60 Hz comes from the power supply and enters the chip at pin 19, TOD; time of day. The 60 Hz signal drives the count in the registers.

It is also an alarm clock. The alarm can be programmed to generate an interrupt whenever it is needed. The alarm is conducted by some other internal registers. These registers are at the same addresses as the TOD registers. The control register for timer B can open up the alarm register. Bit 7 of the control register (Fig. 19-5), if set to a high, allows you to write to the alarm registers and set the alarm. When bit 7 is low, addressing the clock contacts the clock and not the alarm.

Using the real time clock registers is accomplished by careful programming. This is the realm of the programmer and not needed for routine servicing. The servicer should be aware of the clock registers in the CIA and in general what they do.

INTERRUPT CONTROL REGISTER

The interrupts that leave the CIA as *IRQ and go to the processor are vital to the operation of the computer. They alert the processor, and the 8502 then goes into its *IRQ routine. There are five CIA internal inputs into the interrupt register of the CIA. Each type of interrupt is able to set one bit of the ICR.

The internal address of the ICR is HHLH. At the address are two 8-bit registers. There is one register that is contacted when the 8502 writes to ICR. This is the MASK register. The other one, the DATA register, can be reached only if it is being read. Refer to Fig. 19-7.

Inside the CIA, five circuits go to the Data/Mask register combination that is the ICR. Each of the five circuits is assigned its own bit in the Data part of the ICR. An interrupt pulse from any one or more of the circuits will set its individual bit in the Data register.

Each bit in the Data register has a corresponding bit in the Mask register. The mask bit can either let the interrupt through or block it off. When

the mask bit lets the interrupt pass, the *IRQ pin will go low and continue on to the processor. The 8502 will be interrupted and go into its IRQ routine as described in Chapter 12. Should the mask bit not let the interrupt pass, that is the end of the interrupt pulse.

The five circuits and their respective bit positions in the Data/Mask registers are as follows: bit 0 handles underflow from timer A; bit 1 handles the underflow from timer B; bit 2 is used to control the alarm on the time-of-day clock; bit 3 signifies whether or not the serial data register is full or empty; bit 4 indicates the condition of the *FLAG buffer.

Bits 5 and 6 do not have much use but bit 7 is important and tricky. Bit 7 of the Mask register is called SET/*CLEAR. It performs the following mask type job on the individual mask bits. Suppose you write to the Mask register and place a low in bit 7. The register will then place lows into any mask bits that have highs. Bits that already are low will remain that way.

On the other hand, if you write a high into bit 7 of the Mask register, the register will place highs into any mask bits with highs and leave the lows alone.

In order for one of the internal circuits to cause an actual interrupt, first a pulse from the interrupter must enter the Data register and set its bit. The Data register then checks the corresponding Mask bit. If the bit is a low, then nothing happens. Should the bit be a high the interrupt takes another step.

The interrupt will set bit 7 of the Data register. When bit 7 of the Data register gets set, pin 21, *IRQ, goes low. The interrupt is then on its way.

SERIAL DATA REGISTER

At CIA internal address HHLI, there is a complete I/O port. This is the Serial Port, SP. It connects to pin 39. Inside the chip the pin is attached to the SP buffer stage and then onto the serial port register. Refer to Fig. 19-8.

As discussed earlier, the serial port works as a shift register. It connects to the internal data bus and is one of the interrupt circuits that can make

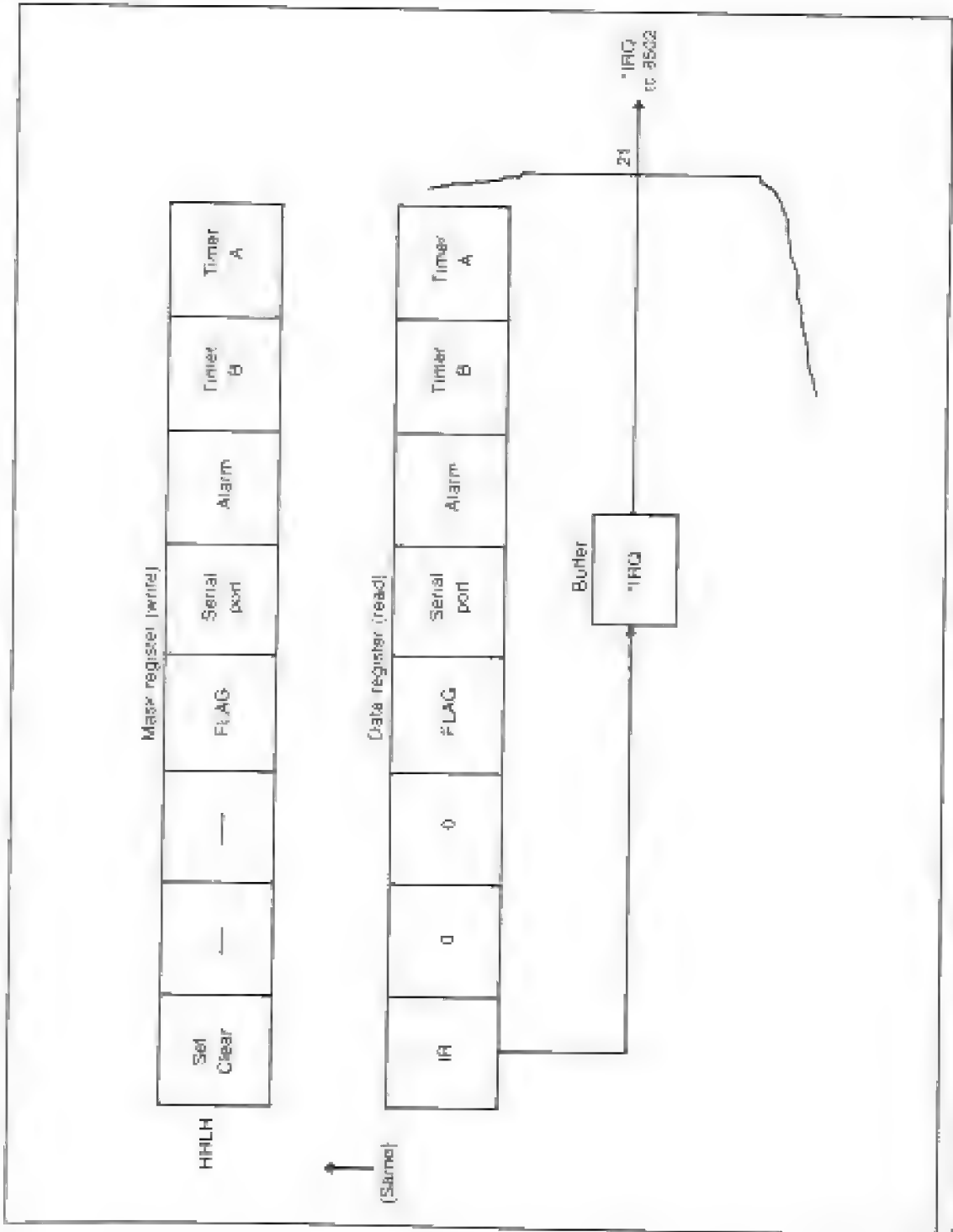


Fig 19-7 The interrupt control register is able to send the low signal 'IRQ' in the 8502

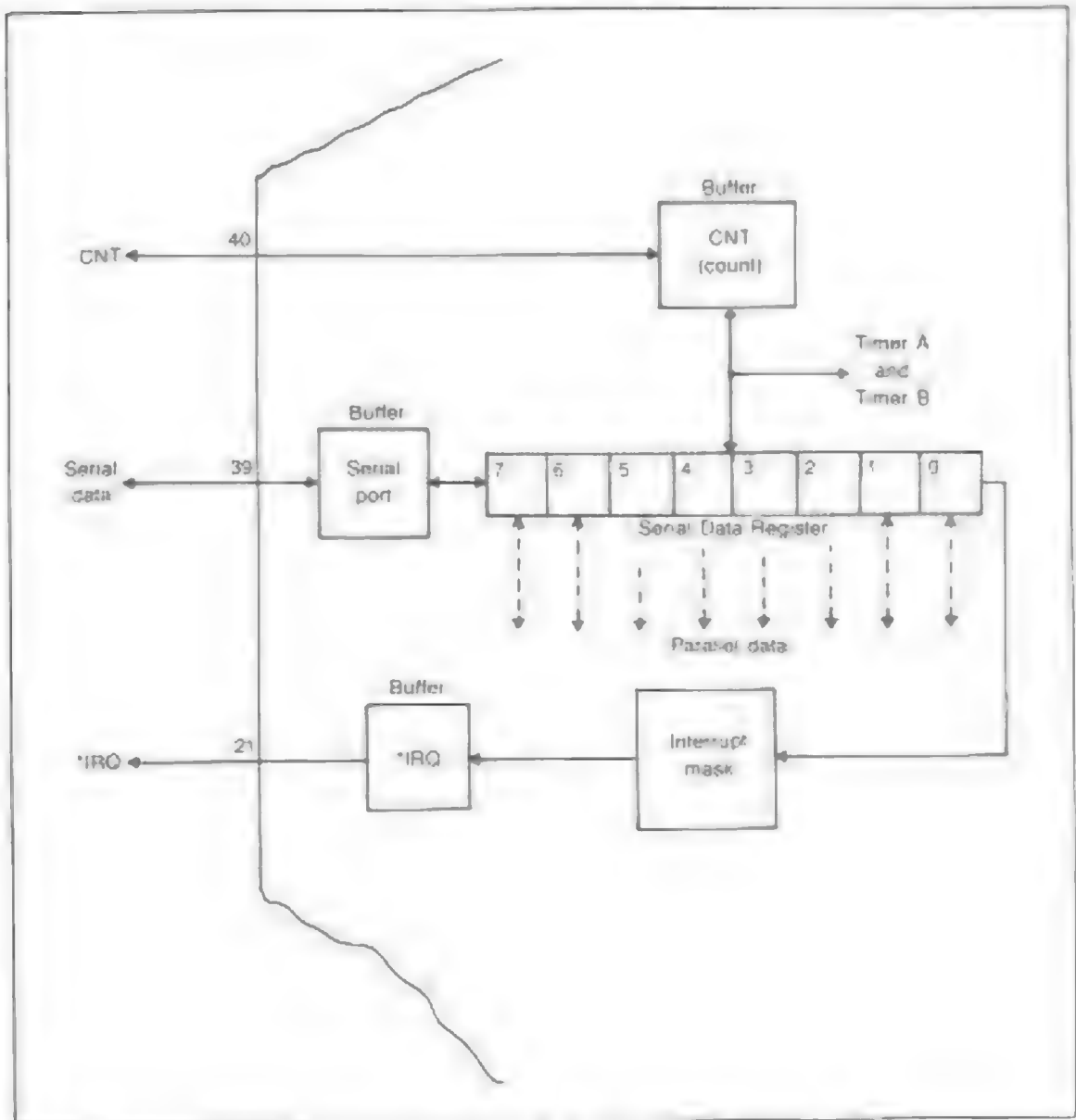


Fig. 19-8 A complete two way serial port is found at pin 39. It works well with *IRQ and CNT.

*IRQ go low. The shift register has its parallel side connected to the eight bit data bus and the serial side to the SP buffer. It also attaches to the CNT buffer and the two timers.

The serial port can be placed into an input or output mode. Bit 6 of the control register A acts

the mode (Fig. 19-5). If a high is put in bit 6, the serial port is an output stage. When a low is placed in bit 6, the port acts as an input.

In the input mode, the CNT pin 40 controls the operation. When there is data coming from a peripheral and is waiting on the SP pin, a rising edge

on CNT opens the SP pin, and the data can enter the shift register one bit at a time. Then after eight CNT pulses the data in the shift register is transferred to the Serial Data Register in a parallel fashion; all of the bits move at the same time. Once the data is in the register, an interrupt is generated. The 8502 takes over from there.

When the port is acting as an output, it needs a timer. Timer A sets a rate of data flow. This will be the rate of flow of the serial bits that leave the SP pin for a peripheral. This computer shifts the data out of the SP pin at $\frac{1}{2}$ the underflow rate of timer A.

The 8502 can output data through the serial port by writing to the serial data register. The only requirement is that timer A is running in the continuous mode. During the write operation the clock signal from timer A is outputted from pin 40, CNT. The data that arrives in the serial register, in time with the clock, is then placed in the shift register and shifted out on pin 39, SP. The data is shifted out starting with bit 7, then bit 6 and so on, ending the byte output with bit 0.

The data is shifted out bit by bit. After eight CNT pulses a byte is outputted to a peripheral. At that time the SP circuit generates an interrupt and sends it to the interrupt Data register. The interrupt is designed to tell the processor that the previous byte has been transmitted and the serial register is ready for the next byte, if there is one.

The processor though runs one step ahead of the SP circuit. If it has more data, it will get the byte into the Serial Register immediately before the interrupt. The data is then shifted out of the SP pin. As long as the processor keeps one byte ahead of the SP circuit, the data output will be continuous. When there is no longer any data, after the eighth CNT pulse, CNT will go high and the transmission will cease.

OPERATION

There are two 6526 CIA chips in the C128. The CIAs do jobs that require a digital-to-digital interface. This is in contrast to the VIC and SID chips that convert digital signals to analog signals. These chips are covered in Chapters 21 and 22.

CIA1

One of the CIAs is used to receive the keyboard input strikes. The same connections that receive the keyboard pulses are also attached to the two control ports. Refer to Master Schematic. Joysticks and other peripheral inputs can also enter here. There is no conflict, under normal circumstances. The keyboard and the joysticks are usually not made to perform at the same time. The peripherals, when off, present a high impedance to the CIA port pins and do not interfere. Even if both were activated at once, there probably will be no real harm done.

The keyboard operates into the CIA port pins in this way. Port B PB0-PB7, pins 10-17, are connected directly to eight rows of keys. Port A PA0-PA7, pins 2-9, are connected directly to eight columns of keys. When the computer starts up, one of the housekeeping jobs that the Kernel does is configure these CIA port pins. It writes to the Data Direction Register of the ports and makes the row connections inputs and the column connections outputs.

There are eight rows wired internally in the keyboard. The wiring bears no resemblance to the QWERTY layout of the keyboard. On each wired row, there are eight columns. At the intersections of the rows and columns, the keys are attached. When you strike a key, you cause a short between a row and a column. There are 64 possible shorts you can cause. The Kernel keeps a close watch on the keyboard to detect any key strikes.

The way the Kernel does this is by scanning the rows and columns continually using the port registers. The decimal address of Port A in the memory map is 56320. The address of Port B is 56321. The columns are input at port A and the rows are input at port B. Bits 0-7 of port A will contain the state of column bits 0-7. Bits 0-7 of port B will contain the status of row bits 0-7.

The Kernel scans the columns and rows in this way. It writes a high to each column in turn. After each column write it reads the status of the rows. If a row and column have a key closure the Kernel will know which column it had written to and which row was shorted. The row and column information denotes one struck key out of the 64 possibles. The

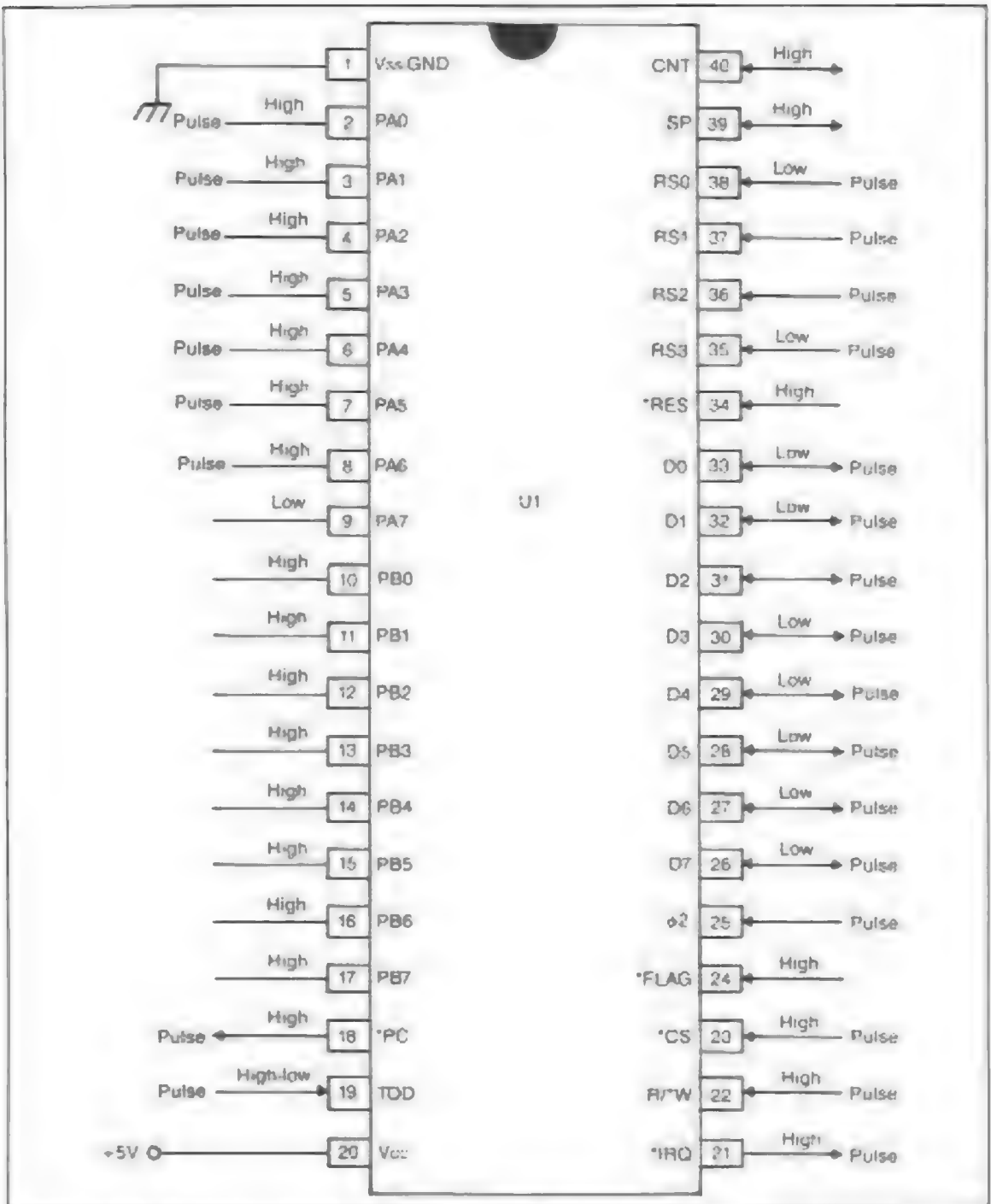


Fig. 19-9 When the computer is first turned on and displays READY and the blinking cursor, these highs, lows, and pulses should be present at U1

Kernel decodes the information and processes the value of the closed key.

While most of the port pins in this CIA are used for the keyboard and joystick type inputs, the serial port, SP and the timing devices in the 6526 are left to do other duties. Pin 40, CNT and pin 39, SP are wired to the User Port. *FLAG is connected to the Serial Bus and the cassette interface.

CIA2

The other CIA uses its parallel output port pins in entirely different types of operations. The port B PB0-PB7 pins are connected directly to the user port pins, C, D, E, F, H, J, K and L. In addition, pin M is connected to Port A pin PA2 and pin B is connected to *FLAG on the CIA. Refer to the Master Schematic.

This arrangement gives a programmer complete input or output control over Port B of the CIA. The additional pins of PA2 and *FLAG permits the programmer to use the port B register in a handshaking operation with a peripheral.

Pin 39 the Serial Port and its companion counter CNT at pin 40 join the other CIAs SP and CNT at the user port plug. The other handshaking line PC also goes to the user port.

PA7 is a data output to the serial bus plug. PA6 is a clock output to the same plug. PA5, PA4, and PA3 also are connected to the serial bus. The serial bus is the place where a disk drive or printer can be plugged in. In this connector, up to five different devices can be connected at one time. The C128 is the controller of the bus. Each device is given a bus address. The addresses are numbered from 4 to 31.

A programmer with the aid of the port A pins, PA7-PA3, is able to control, talk, and listen to any

and all devices connected to the serial bus. Only one device can talk at a time, although all of the devices can listen all the time.

TESTING

The two CIAs perform a lot of complex I/O jobs. You can test to see if the CIAs are set up to do their duties by probing the CIA pins one by one with the logic probe. The test point charts in Figs. 19-9 and 19-10 show what state should be present on each pin when the computer is first turned on and displays the READY message and the blinking cursor. It is not necessary to know if you are reading a port pin or a control state; it is only required that you compare the voltage or logic state you find with what should be there during normal operation.

If you should find a test point that has a reading that does not match up with what should be there, that could be a valid service clue. Then it is useful to know more about the test point with the discrepancy. At that time, you find out what pin it was that has the wrong reading. Then if you have an idea of what the pin is doing and how the circuit involved is operating, you can intelligently come to some conclusions as to what could possibly be causing the trouble.

Most of the circuits in the CIA area of the computer have been buried in the CIA chip. There are no simple and easy ways, besides direct replacement of a chip, that will pinpoint a circuit trouble. Your technique must be an analysis of the input and output signals and states. The results of the tests can give you an idea of whether a state is entering properly, was processed and if it is exiting correctly. Your understanding of the workings will allow you to come to such a conclusion.

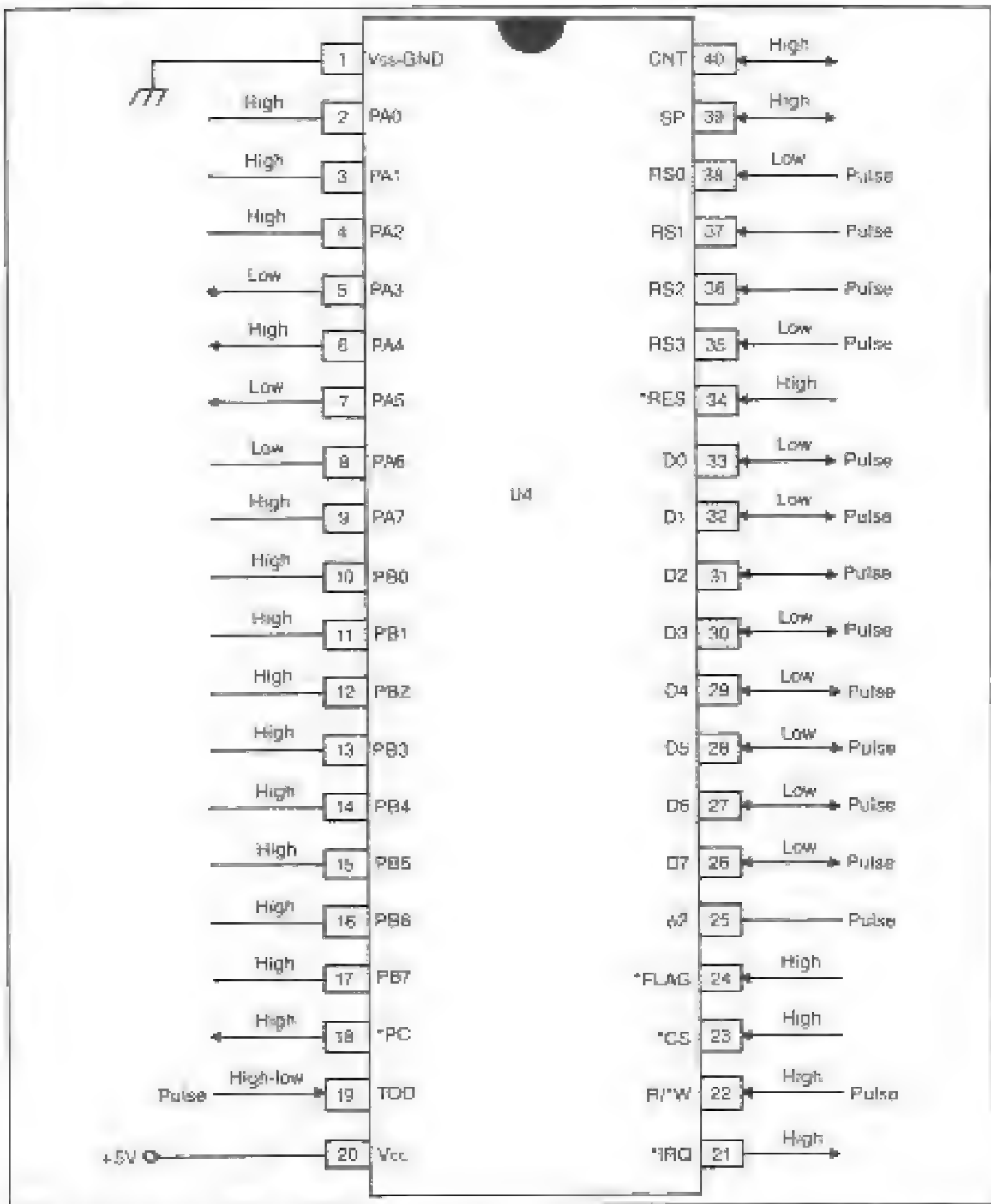


Fig 16-40 Even though U4 is identical to U1 in construction, they are connected in different circuits and do not have the same signals on corresponding pins.

20. 8564 Video Interface Chip

TV Video displays for computers are relatively new. In the days before microcomputers, the mainframes and mini's used teletype machines as a keyboard and printer. The keyboard was the important input and the printer the main output. There were some forms of video display systems but they were used mostly for special purposes.

When personal microcomputers arrived, they came with TV displays. At first, the video interface circuits consisted of a number of chips that performed the step-by-step conversion of digital bits to analog video signals that a TV device could use. Then as the popularity of TV displays became apparent, all sorts of video output chips were designed and produced. Commodore became known for its VIC systems, the Video Interface Chips. In the C128 is one of the latest in the line.

The 8564 VIC is an upgrade of the 6346/6367 VIC used in the C64 machine. It contains all the circuits needed to keep it completely compatible so that the C128 can output during the use of its built-in C64

mode. The 8564 also has a number of additional features to let it work with the C128 mode too.

The older VIC is housed in a 40-pin DIP. This newer 8564 VIC, seen in Fig. 20-1, has 48 pins so it can input and output the extra features. The most obvious additional features of the new VIC are K0, K1 and K2, the handlers of the extra keys needed in C128 mode, and the new clocks for 2 MHz and Z80 operation. The VIC still generates the 1 MHz clock, plus producing all the various graphics used in 40-column mode. The 80-column mode is produced by the other video output chip, the 8563 covered in the next chapter.

VIC GRAPHICS

The word "graphics" includes every type of output that VIC can produce for display. First of all are the alphanumerics, the alphabet, numbers and symbols that appear as you type on the keyboard. Then, the drawings, custom defined characters and sprites are output that the C128 is able to develop.

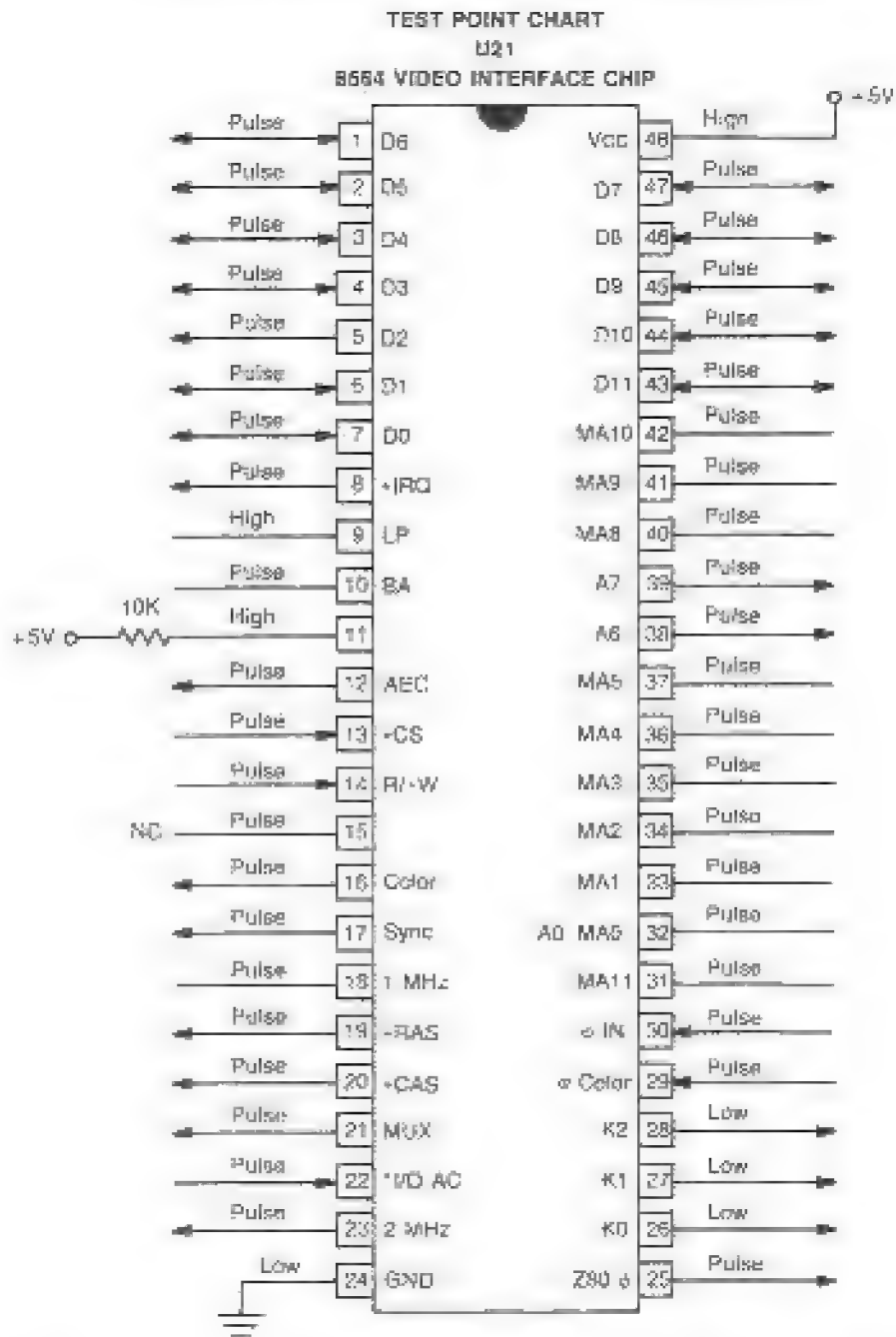


Fig. 20-1 The 48-pin VIC has digital inputs and outputs as well as analog outputs.

All these items are graphics. As an example, let's take an overview of what happens when you strike a key and it appears on the screen.

The keystroke shorts out a row-column switch in the keyboard and generates a specific set of bits that passes through the CIA, goes to the processor, and is stored in video RAM. In standard color character mode, 1000 bytes of RAM, between addresses 1024 and 2023, are assigned to hold the codes for the characters that are to appear on the screen. This address space between 1024 and 2023 is called video RAM.

Up on the TV screen, in the display block, there are 1000 character blocks, 40 columns across and 25 down. The video RAM matches up with the 1000 character blocks on a one-to-one basis. Each character block consists of eight rows and eight columns. Each row contains eight dots of light. Each dot can be turned on or turned off by VIC. This makes each character block contain a light matrix of 64 dots.

Stored in the video RAM are bytes of code, one for each character that can be displayed. To actually convert the video RAM code stored to a character is a chore that VIC and the character ROM take care of. In the character ROM are the actual bits to display the character. Each character takes up eight ROM bytes. Eight ROM bytes make up a bit matrix of 8 bytes \times 8 bits: a 64 bit matrix. If a bit is a high it will light a dot on the screen. Should a bit be a low it will turn off a dot of light on the screen. VIC takes the code from RAM, and according to the code, takes a character from the ROM and displays it.

VIC scans the Video RAM 60 times a second and picks up all the latest character codes. It then places the current characters into their screen character blocks. VIC is continually scanning video RAM and constantly updating the TV screen. You can walk away from your working computer and VIC just keeps on scanning RAM and updating the screen.

VIC performs the same type of scanning and updating on the screen in its other modes too. A second graphic mode is called standard high resolution. Instead of using 1000 character blocks, VIC uses the entire display block as one. In the block there were

40 characters across with eight dots of light for each. $40 \times 8 = 320$. In each block there were 25 characters down. With eight dots in each block down, $25 \times 8 = 200$. This gives the C128 a high resolution of 320×200 . The individual dots of light are the picture elements, nicknamed pixels.

Another mode for the C128 graphics is called split screen high resolution. This is quite like the usual high resolution except a character window can be installed at the bottom of the screen. This provides text with the graphics.

The above high resolution modes allow you to program in two colors. Another mode lets you place four colors into the picture. However, there is no free lunch, the resolution is reduced to 160×200 pixels. This mode also has a companion split screen mode to give you the option of having a text window at the bottom of the picture too.

Besides the VIC type modes, there are more video capabilities in the C128 out of the 8563 Video Controller chip. These are discussed in the next chapter with the 80-column capability.

VIC PIN-BY-PIN OPERATION

VIC has addresses: in decimal from 59248-59296. When AEC out of pin 12 is high, as in Fig. 20-2A, these registers can be contacted by the processor through address lines A0-A5. There are only 49 registers, and six address lines are able to contact 64 addresses. Lines A0-A5 do the job easily. Lines A0-A5 arrive at pins 32-37 and when the processor is using them they are inputs to VIC. The logic probe should show them all with pulses. AEC will show a pulse too.

During the low of AEC, as Fig. 20-2B shows, the processor turns control over to VIC. VIC has a complicated multiplexed address situation. VIC has to keep addressing the video RAM to keep updating the TV display. VIC uses pins 32-39 for the multiplexing of the video RAM addressing. When VIC is using these pins they are outputs. First it addresses the rows and then it addresses the columns in video RAM. The multiplexing allows the pins to output 14 address lines, eight for rows and six for columns. The 14 address lines are able to contact 16K addresses. The system RAM of 128K is ar-

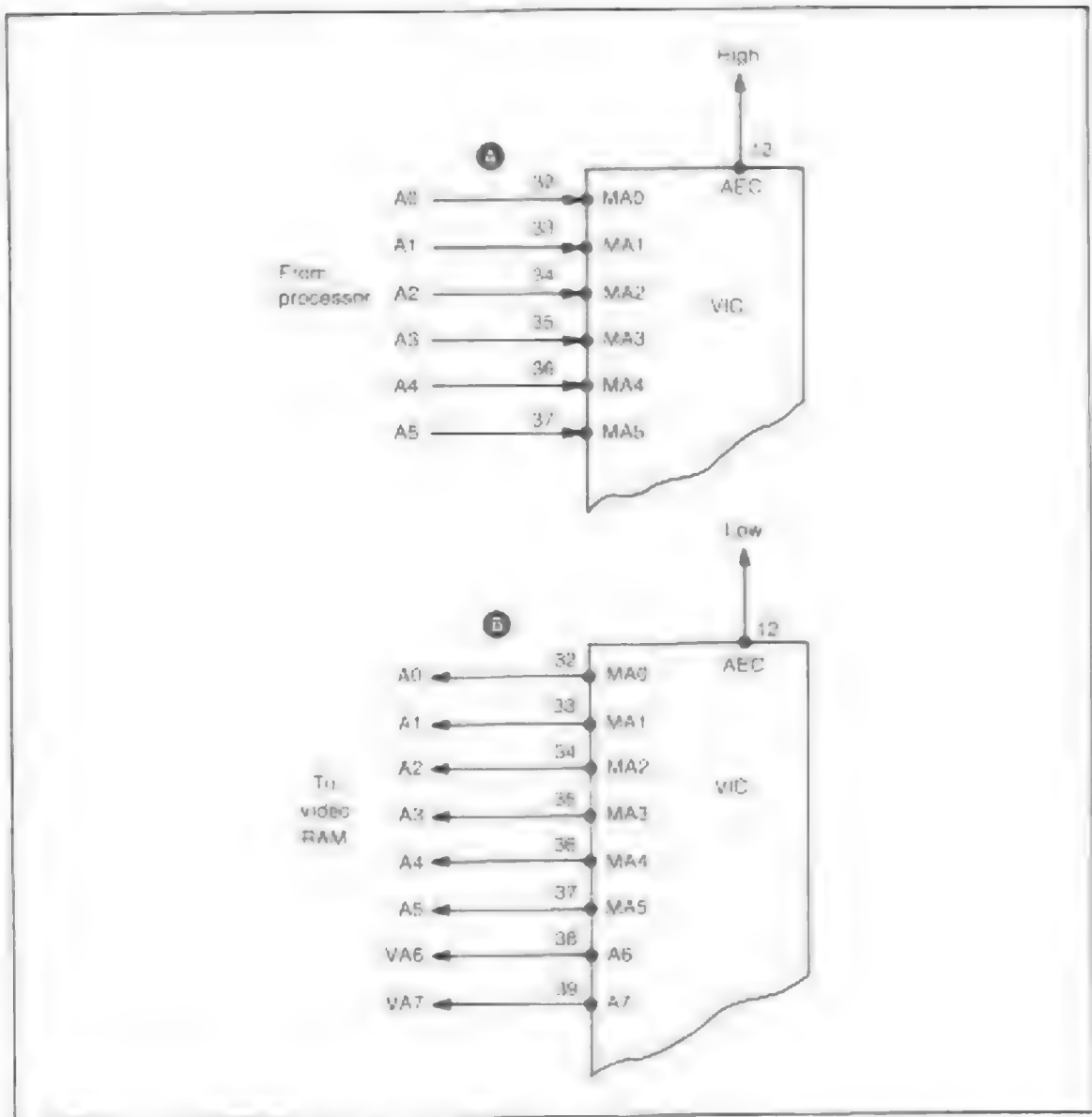


Fig. 20-2 The processor can address VIC through its six-bit MA5-MA0 multiplexed address pins. VIC, in turn, uses these same pins to send bits the other way to address video RAM.

ranged in eight 16K video banks. VIC can address any one of the banks. The video RAM can be installed in any desired bank. These address pins should read pulses on the logic probe.

Pins 31, 40, 41, and 42 are more address lines, MA6-MA11, in Fig. 20-3. These are also outputs

from VIC. They are used by VIC to address the character ROM and the color RAM. The pins should all read pulses on the logic probe.

Pins 7-1 and 43 are connected to the system data bus, D7-180. They are only for use by the processor when it accesses the VIC registers. They

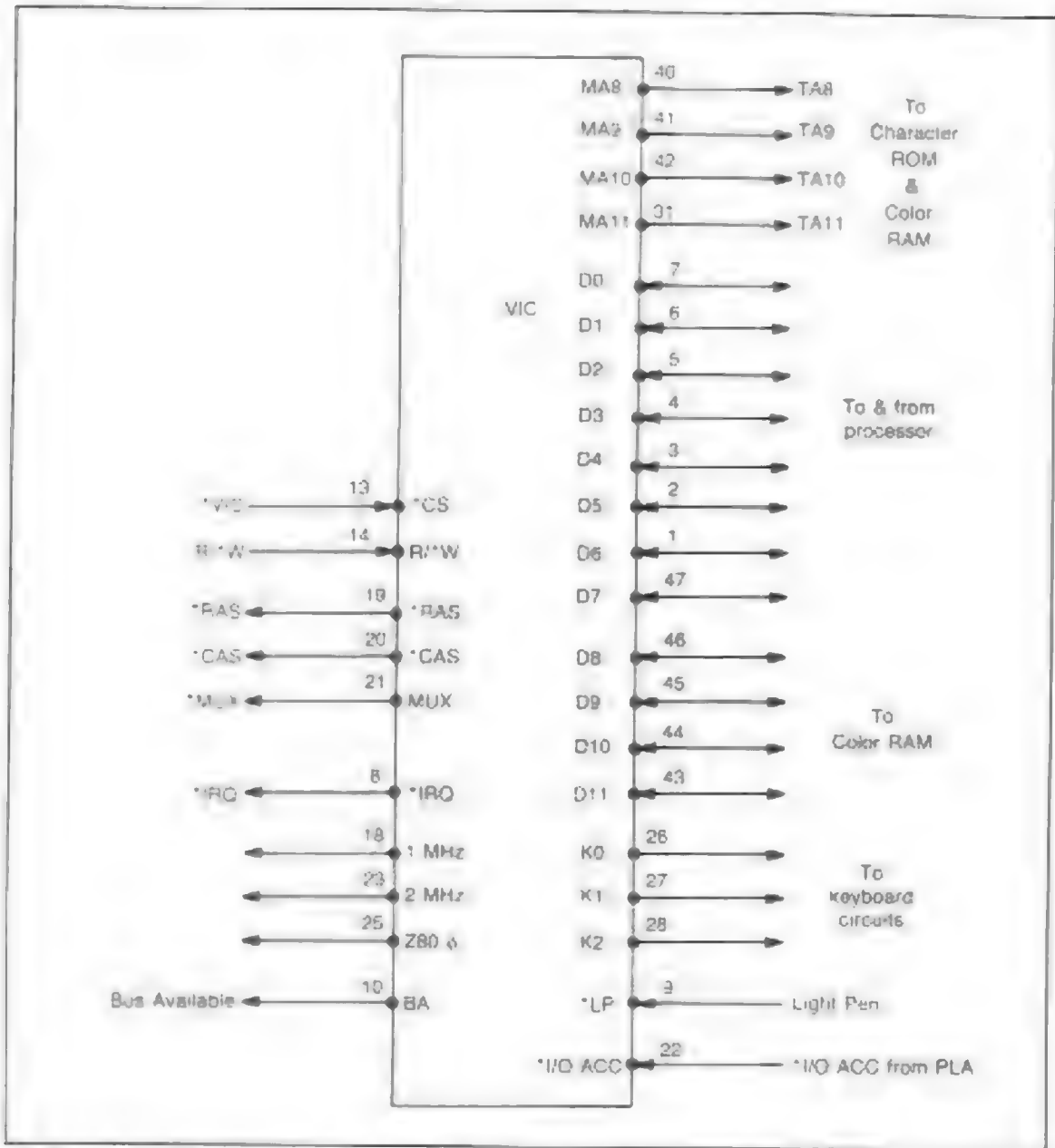


Fig. 20-3 VIC has more address lines, MA8-MA11, to access the character ROM and the color RAM.

are two-way lines so that the processor can both read from or write to the registers. They work only when AEC is high.

More data lines come from VIC. They are at pins 46-43 and are called D8-D11. These are spe-

cial data lines used only by VIC to access the color RAM. Color RAM can only output a color nybble over four lines. These color nybbles have a one-to-one relationship with the video RAM. When VIC gets a byte from video RAM it also gets its corre-

sponding color nybble for the same character block from color RAM. The four lines can provide one of 16 colors for the character block. The four lines should all show pulses too.

The following are examples of control lines that enter and leave VIC. Pin 13, a pulse from the processor to select the VIC chip, is called *CS. Pin 14 is the R/W line from the processor that provides the control that decides the direction the data is to flow over the data bus between the VIC and the processor. The *RAS line is out of VIC's pin 19 that provides the row address strobe for system RAM. *RAS has a companion signal called *CAS, from pin 20, that contributes the column address strobe for system RAM.

Next out of pin 21 comes the *MUX multiplexing control for the RAM chips. Out of pin 8 emerges the *IRQ interrupt signal from VIC that lets the processor know an interrupt is happening. Finally the clock pulses leave their origination circuits in VIC. From pin 18, the 1 MHz clock leaves VIC. The 2 MHz clock comes out of pin 23 and the 4 MHz Z80 clock emerges from pin 25.

When you test all these pins just mentioned in this paragraph with the logic probe, and they are operating normally, they should all read as pulses on the probe. If the pulse is missing on any of the pins, that is a signal of trouble. Should the missing pulse be a VIC output, you probably have a bad VIC chip. A missing input pulse indicates trouble in the circuits that are supplying the pulse.

Pins 26, 27 and 28 are K0, K1 and K2. These three pins lead to the Keyboard Control Register in VIC: bits 0, 1 and 2. The register is at decimal address 53295. The register continually scans the keyboard to see if the three keyboard lines found only in C128 mode, but not used in C64 mode, have been pressed. This permits the keyboard to be able to be used in both C64 and C128 modes.

If you read the three pins with a logic probe, they will all show LOW. The three pins connect to the three lower bits of the register. The other bits, 3-7, are not used in the C128 so far. They do not connect to pins. You can read them with the Monitor MEMORY command or with a PEEK. If you do they will be found to be held high.

Pin 9, *LP, is reserved for a light pen input. The light pen latches the screen position of the spot the pen is touching into two registers, 53267 and 53268. They are Light Pen X and Light Pen Y. The logic probe will read this pin as a high.

Pin 10, BA, is the bus available VIC output. It goes to the RDY pin on the 8502, the *BUSRQST on the Z80 and the BA pin on the PLA. When VIC outputs a low to the RDY input at the 8502 and the *BUSRQST input on the Z80, the processor that is on will finish its current operation and then shut down. This makes the system bus lines available for sharing and direct memory accesses. Pin 10 will read a pulse on the logic probe.

Pins 11 and 15 are unused. Pin 11 will read a high because it is tied to +5 volts through a 10K resistor. This keeps it out of the way. Pin 15 is not connected externally. It will probe a pulse from its internal connections. Pins 48 and 24 are a +5 volt source and ground.

Pin 22 is an input from the PLA. It is called *IOACC. It tells VIC that an I/O access is taking place, and VIC should accommodate the access by stretching the clock. The logic probe will read pulse.

Pins 29 and 30 are the two connections to the clock chip discussed in Chapter 17. Pin 29 is the color clock input to VIC, as in Fig. 20-4. From the color clock the Chroma signal is formed. The Chroma signal leaves VIC at pin 16 and goes to the Chroma input of the RF Modulator. Pin 30 is the Dot clock input from the clock chip and is the frequency that all the rest of the system frequencies are derived from. These include the 1 MHz, 2 MHz and 4 MHz clocks. In addition, the output from pin 17, Sync/Luminance, comes from the Dot clock too. The pin 17 output goes directly to the RF Modulator too. All of these pins will show pulses on the logic probe under normal operating conditions. These pins are also good places for other tests, which are covered later in this chapter.

VIC Character Fetch

VIC has 49 registers in the C128. This is two more than in the C64. The additional two registers are used in the C128 to handle the three additional keyboard lines at 53295 and to set clock frequen-

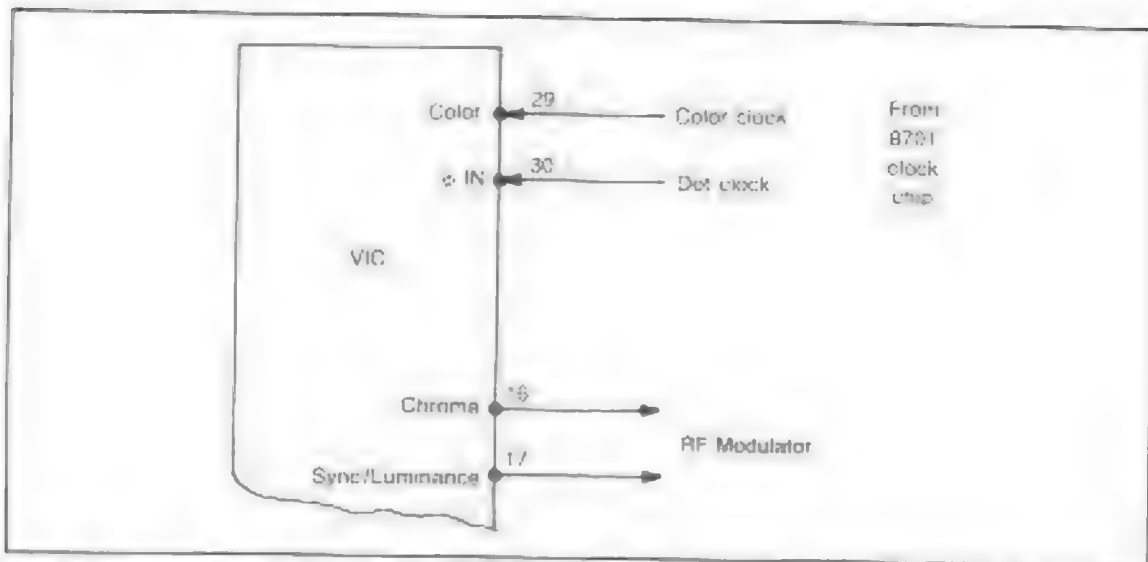


Fig. 20-4 VIC has inputs from the clock chip that set the frequencies it will be running at and outputting. Two such outputs are Chroma and Sync/Luminance that produce the color TV display.

cies at 53296. Otherwise the VIC registers are about the same in both the C128 and C64.

The start address for VIC is 53248: register 0. The last address is 53296: register 48. For a character fetch, register 24 (53272), the VIC memory control register must be contacted with a program line. The data to form the video RAM and character ROM addresses are in register 24. Bits 7-4 contain the video matrix base address and bits 3-0 have the character dot-data base address.

Video RAM must be addressed by VIC to obtain the character pointer code bits. VIC uses 14 multiplexed address bits to access video RAM. The 14 bits will address 16K of the total RAM. The RAM is arranged in 16K banks so that VIC can access with only 14 bits. All of the chips that VIC needs, such as the video RAM, color RAM and character ROM, are arranged so they appear in the 16K bank that VIC will use to produce the display character.

VIC begins to put together the address of the character pointer with the aid of register 24. As in Fig. 20-5, the highest four bits of 24 are the address bits A13-A10 of video RAM, VIC outputs them as part of the address of the character in ROM.

Meanwhile, internal to VIC is a counter register that has ten bits and is constantly counting from

0 to 999—which is the 1000 video RAM locations and the corresponding 1000 TV character blocks on the screen. The ten bits from this counter and the four bits from register 24 form the address for the character pointer. The counter is constantly scanning the 1000 consecutive places. When a desired character is needed by VIC, it outputs the ten additional bits through MA0-MA9 of its address pins. The video RAM is accessed in this manner and gives up a copy of the eight bits in the pointer address.

The next step is to form another 14-bit address so that VIC can access the character ROM for the eight bytes of dot information to form the character in lights on the screen. The makeup of the address is shown in the bottom register in Fig. 20-5. The three most significant bits of the address are found in register 24 in bits 3, 2 and 1. VIC uses them as address bits A13-A11 to point to the character ROM. The next eight bits of the pointer, A10-A3, are the bits VIC has just fetched from video RAM. That leaves only A2-A0 to be filled in order to form the character address that will let VIC access the character ROM.

To review, bits A13-A11 are the character ROM select bits. These three bits are found in VIC's own register 24, bits 3, 2 and 1. Bits A10-A3 are

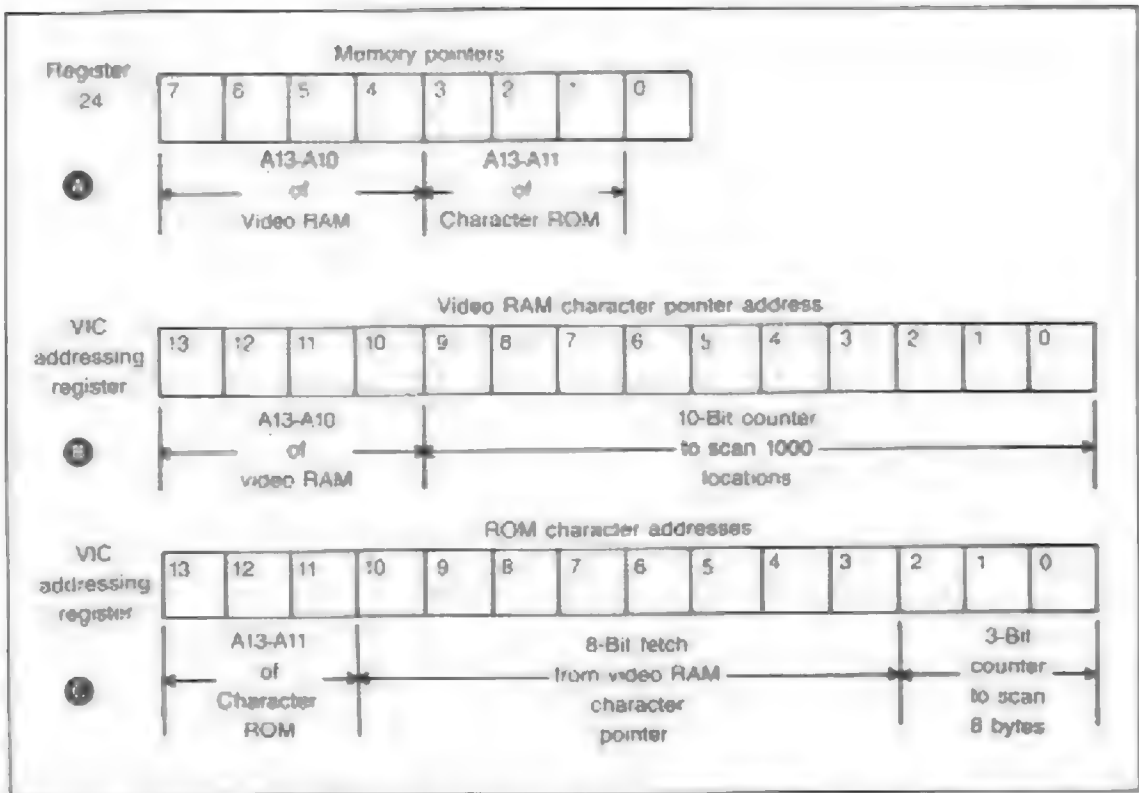


Fig. 20-5 In the highest four bits of register 24 are address bits A13-A10 of video RAM (A). They are combined with a ten-bit counter that provides the A2-A0 bits. VIC outputs the 14-bit address, accesses RAM, and has a character pointer returned (B). In bits three, two, and one of register 24 are address bits A13-A11 of the Character ROM. A10-A3 are taken from the character pointer just received from video RAM. A2-A0 come from a three-bit counter in the VIC (C).

the register select to choose from among the 256 characters in a set in ROM. That leaves A2-A0.

A2-A0 is a three-bit counter. The three bits can count from 0 to 7 in binary: 000, 001, 010, 011, 100, 101, 110 and 111. There are eight bytes to one character. At each ROM byte location, the counter points to each byte in turn till all eight bytes of a character have been addressed and accessed by VIC.

In the character display modes, the activity can be summed up quickly. VIC starts the memory accessing by first reading a character pointer out of video RAM. The pointer is one byte in width and is an address. The address is the start location in the character ROM where the desired character is stored in eight bytes. The counter continues to address all eight bytes. The 64 bits in the eight bytes

that contain the character in highs and lows are processed by VIC to light up the 64 dots in one of the 1000 character blocks to form the character. The highs light their respective dots while the lows extinguish the dot lights.

Character Colors

In the color RAM chip that attaches to VIC through four data pins, the registers are arranged to have four active bits. The other four bits are disabled. These nybbles control the color of the character that gets displayed. Because each register has four usable bits and four bits can have 16 combinations, each nybble can code one of 16 colors.

The color RAM has 1000 of its registers assigned on a one-to-one basis to the 1000 bytes in video RAM. The wiring is also arranged so that

when a video RAM location is addressed, its corresponding register in the color RAM is also addressed at the same time. It is as if the video RAM has 12 bits instead of its eight. Eight bits for video and four bits for coloring.

As the 12 bits are thus addressed, the eight video bits enter the data bus D0-D7 and enter VIC pins D0-D7. The additional four bits from the color RAM leave the chip and enter VIC at pins D8-D11. VIC then processes all 12 bits.

Character Modes

Characters are displayed by VIC in three different modes. One mode, the Standard Character mode is the one that comes up automatically when you turn on your Commodore 128. The operating system sets up this mode during its housekeeping duties.

The VIC will adopt a specific character mode as three bits in its registers are set or cleared. The

VIC is designed to respond with a mode as the following three bits are affected either by the operating system or by planned programming: The first bit is named MCM and is in register 22, bit position 4. The second bit is called BMM and is found in register 17, bit position 4. The third bit is called ECM and is also in register 17 but at bit position 5. Refer to Fig. 20-6.

When your C128 is first turned on, all three bits are cleared and hold lows. This forces VIC to adopt the *Standard Character mode*. In this mode, when eight bytes of a character are fetched by VIC, all eight bytes are displayed directly onto the eight lines of each character space on the screen.

If you place 0's into these bit positions the background will be displayed as a result of register 33. When 1's are installed, the foreground color is selected by the color nybble. This is performed as desired by the programmer.

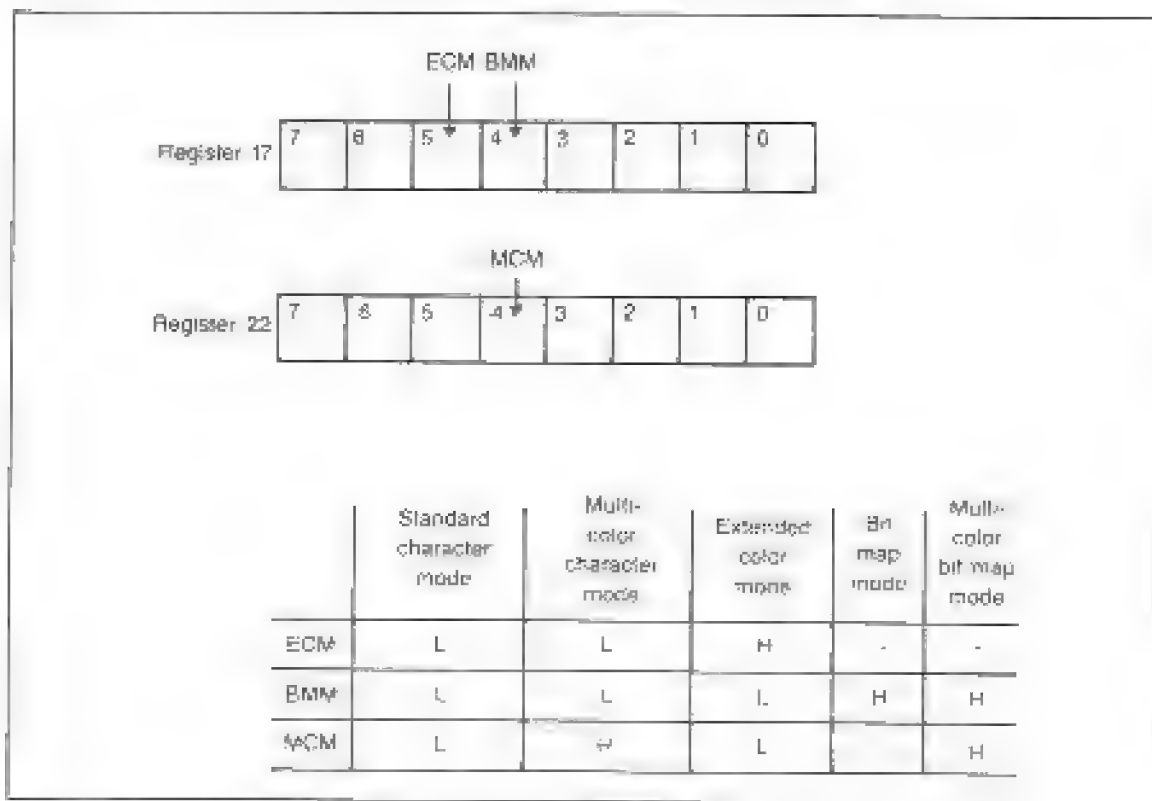


Fig. 20-6. The three bits that set up the various VIC modes are found in registers 17 and 22.

The *Multi-Color Character* mode is obtained by installing a 1 in MCM. BMM and ECM are left low. This mode allows the programmer to have up to four colors in each character space. However the resolution of the character suffers with the additional coloring. Two bits are needed to specify one dot color in this mode. This reduces the 8×8 dot matrix to a 4×8 matrix with each dot twice its original horizontal size. The character is not quite as distinctive as in the 8×8 matrix.

The *Extended Color* mode is produced by setting ECM to a 1 and leaving BMM and MCM as 0's. This mode lets the programmer have individual selection of background colors for each character space in the 8×8 matrix.

The extended color feature is paid for in the number of characters that the VIC can use in this mode. Only 64 characters are available because two of the character address bits are being used for the color information. For further details on programming these three character modes, refer to a programming book that covers the VIC from this point of view.

Bit Map Modes

Bit 5 in the VIC register 17 is BMM, Bit Map Mode. If the bit is set to a 1 the BMM is on (Fig.

20-7). When the bit is cleared to a 0 the mode goes off. In the character modes, VIC accessed video RAM to get a character pointer. This was an address to eight bytes in ROM that was storing a character. The character was installed in 64 light dots in a screen character space.

In the bit map mode, the VIC displays in the same 64 light dot spaces but has more detailed control. The VIC, in bit map mode, is able to control each of the 64 light dots by itself. A bit in video RAM is assigned to each and every light dot on the screen. If the bit is a 1, the dot goes on. When its assigned dot is a 0, the bit goes off.

There are 1000 character spaces on the screen. There are 64 light dots in each space. This means there are 64000 dots on the screen to be controlled. If one video RAM memory bit controls one dot, there has to be 64000 bits in video RAM to handle the screen. Eight bits to a byte means the bit map mode requires 8000 bytes of memory for the control.

The resolution of the mode is 320 horizontal dots by 200 vertical dots. The VIC accesses the 8000 bytes of video RAM in bit map mode in the same way it accessed the 1000 bytes of video RAM in the character mode. However, the VIC is not looking for character pointers. It needs direct color data.

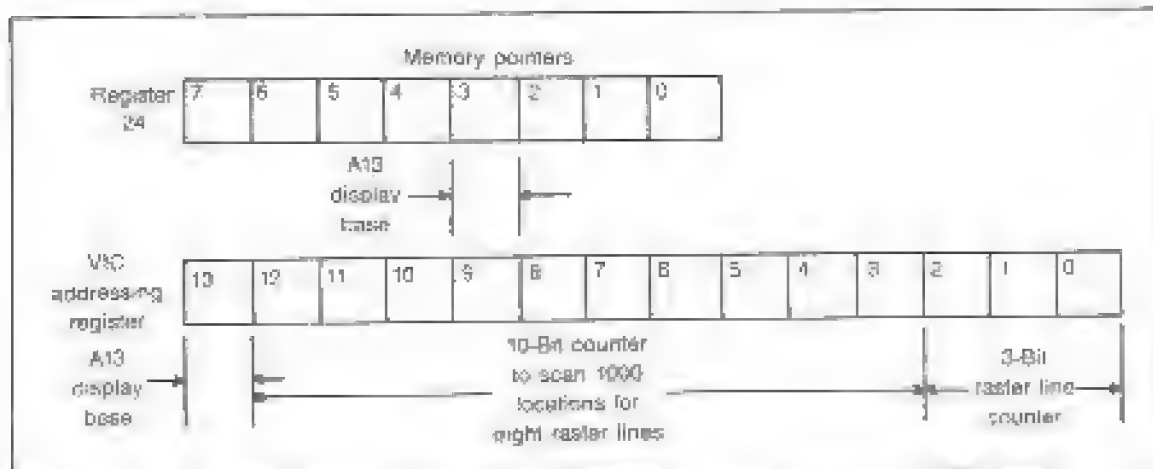


Fig. 20-7 In bit map mode, the VIC latches data from memory on a one to one basis where the character type latching bit three of register 24 is used as A10 in this mode. The rest of the address is formed with the ten-bit counter providing A12-A3 and the three-bit counter contributing A2-A0.

The address that the VIC forms to access video RAM is also produced by a counter. The counter constantly scans the video RAM to keep the display up to date. A13 of the address VIC outputs is taken from register 24, bit 3 called CB13. Refer to Fig. 20-7. Bits A12-A3 are the outputs of the video RAM counter. The last three bits A2-A0 are the counter that covers the eight lines in each character space.

The addressing of the 1000 spaces on the screen is quite like the way the character pointers are addressed. A13 is the chip select for the video RAM start location. A12-A3 scans the 1000 spaces: it scans the 40 horizontal spaces eight lines at a time. A2-A0 then counts the individual eight lines in each space. That way each dot is individually addressed.

When BMM is a 1 the Standard Bit Map mode appears courtesy of the VIC. The color information is obtained only from the state of the individual dots in video RAM. The Color RAM chip has no effect in this mode.

The color of dots in one of the 1000 spaces is controlled by the screen memory byte of the space. For instance, decimal 1024 is the screen memory address of the upper left hand space on the screen. In bit map mode, this pointer becomes the color controller of that single block. There are eight bits in the memory pointer.

The bits are arranged in four more significant bits and four lower bits. The four bits each become a color controller. In bit map mode, the higher four bits are the color code for all the bits in the block that are set as 1's. The lower four bits are the color code for all the bits in this block that are reset to 0.

The *Multi-Color Bit* map mode is produced by not only setting BMM to 1, but also MCM, bit position 4, in register 23 to a 1. The additional feature of more colors has a price to be paid too. Two bits are used to select the colors and the size of the horizontal dot has to be doubled reducing resolution to 160 horizontal dots by 200 verticals. However, three separate colors plus the background color can all be displayed in any of the 1000 8 x 8 dot spaces.

SPRITES

One of the exceptional abilities of the VIC is the manufacturing of sprites. The engineering spec-

sheets on the VIC call them MOB's for movable object blocks. The MOB is a character. The MOB is not found stored in a ROM like the keyboard characters. MOB's are conceived and designed by you. Once designed they are stored in RAM locations.

A MOB character is seen on the screen in a 24 x 21 dot arrangement. This is much larger than the 8 x 8 dot characters one of the 1000 spaces can display. A MOB needs 63 bytes in memory in comparison to the eight bytes that a keyboard character requires.

The VIC can display up to eight MOB's at one time. The large characters can be placed at any spot on the TV screen. They can be made in color. They can be magnified and moved around. The ability to use the MOB with its graphic capabilities makes the VIC an exceptionally good chip to display arcade type games. For further MOB programming details, there are a lot of books on the subject.

Once a MOB is programmed, the dot information is stored in 63 bytes of memory. Each three bytes of consecutive memory defines one line of the character. Three bytes contain the dot light information for 24 spaces. Since there are 21 lines in a MOB, the complete character requires 63 bytes to turn all the dots on and off.

To find a MOB in RAM, the VIC must output the correct 14-bit address. It forms the address with an 8-bit pointer that locates the start address of the 63 byte character and a six bit counter that steps through the 63 bytes. Refer to Fig. 20-8.

VIC gets the pointer from video RAM. It produces the counter from an internal register. If you look at the VIC's register map, there are 16 MOB x and y position registers (Fig. 20-9). One x and one y register for each of the eight sprites. The VIC uses these positions with reference to the upper left hand corner of the TV screen to locate the MOB on the screen.

For the actual locating, VIC considers the TV face as having a resolution of 512 horizontal positions and 256 vertical positions. This includes the entire screen; border as well as display area. The visible screen area is from positions 23 to 347 horizontally and 50 to 249 vertically. With the x and y positions from the MOB registers for each of the

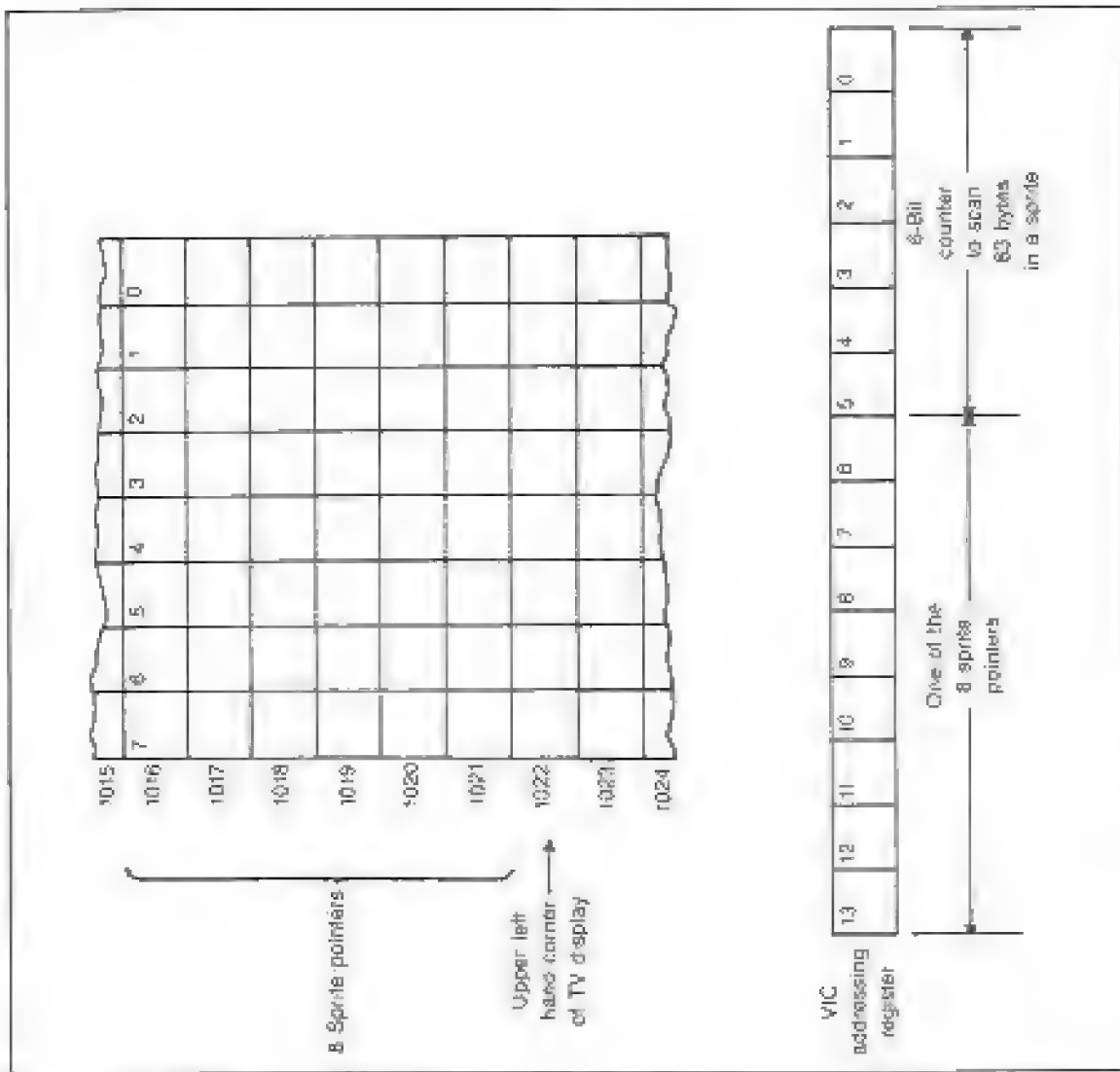


Fig. 20-8. In order to locate a MOB, the VIC must output an address consisting of one of the sprite pointers on A13-A6 and a six-bit counter in A5-A0.

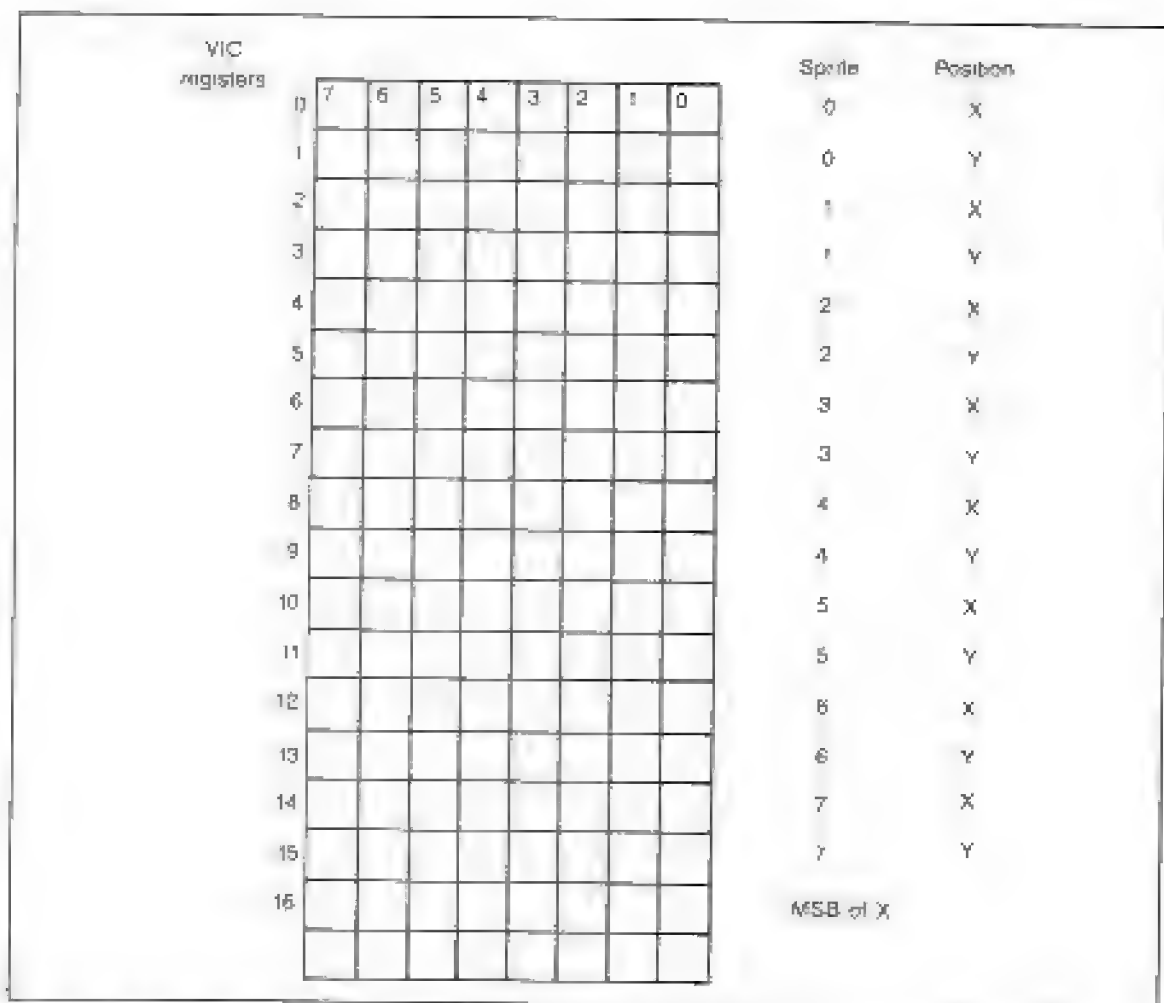


Fig. 20-9. Sixteen of VIC's registers are used to locate a sprite on the TV screen.

right sprites, the VIC can easily install the dot information on the TV face.

Register 21 in Fig. 20-10, in the VIC is the off-on-switch for the sprites. There is one bit for each MOB. If the bit is set to a 1 the sprite will light up. A 0 will turn it off. Register 23 and 29 perform the magnification of a sprite. Register 29 expands it horizontally and 23 expands it vertically. A 1 does the expanding while a 0 makes it a normal size. If you want to display one sprite on top of another, display register 27 will do the job. Just install a 0 in the MOB's bit. A 1 in the bit will shift the display priority to the original display.

There are all sorts of tricks you can do with sprites, but here again we are getting into the realm of the programmer. It is important to comprehend all these workings of the VIC for servicing in case a feature fails. If it does, you will be able to pinpoint where the trouble is by knowing how the system is supposed to be operating normally.

OTHER VIC FEATURES

The VIC registers have some other jobs that they are assigned to perform. One such job is the ability to blank out the screen. In register 17, bit 4 is named DEN. It is usually set to a 1 which is

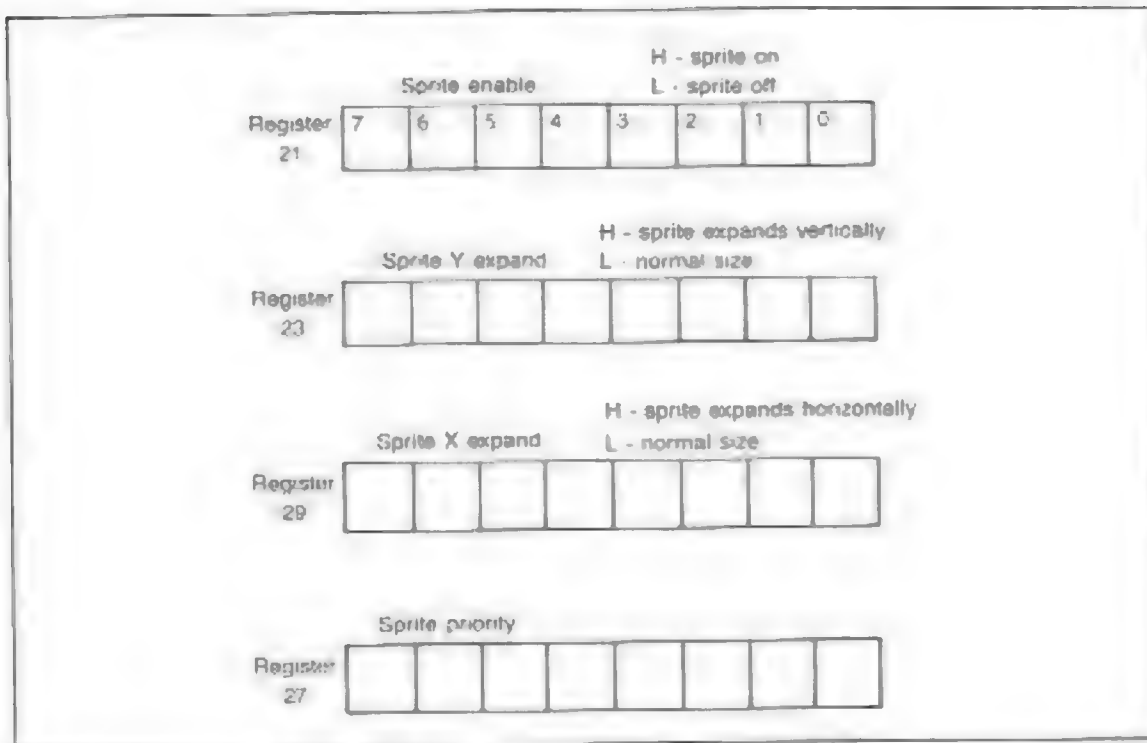


Fig. 20-10 Register 21 contains eight off-on switches, one for every possible sprite. Register 23 expands the sprites vertically while 29 expands them horizontally. Register 27 can place a sprite on top of any other display that might be on the screen.

the normal display mode. If you install 0 in the bit, the screen will be blanked out. When it does blank out it will assume the color prescribed by the bits in register 32, which sets the exterior color.

The usual display pattern on the screen is 1000 character spaces with a 40 x 25 layout. Bit 3, CSEL in register 22 and bit 3, RSEL in register 17 is normally set with 1's to produce this arrangement. If you want to change this to 38 x 24 install 0's into the bits.

The VIC provides either vertical or horizontal scrolling so game characters can be easily moved around the screen as programmed. Register 22, bits 0, 1 and 2 moves the display data an entire character space at a time in the horizontal position. Register 17, bits 0, 1 and 2 moves the display vertically.

Raster Register

Register 18 is the raster register. The raster, of course, is the lines of light on the TV screen. The

VIC places the display precisely onto the lines of light. In order to do this, the VIC must know which dot of light is being lit or turned off at every instant of time. VIC is aware of the dot timing.

The raster register keeps track of the current raster position. If you read the register, the lower eight bits of the position are found in the register. The most significant bit of the position is then found in register 17, bit 7, RCB. This data is used by programmers to make timing changes in the display to get rid of flicker. The changes are made while the border is being scanned by the CRT cathode ray and not during the display window. The changes are made before the cathode ray reaches position 51 or after position 251.

While the raster register read is useful to eliminate flickering in the picture, a write to the raster register, including a write to bit 7 of register 17, performs another job. The data written gets latched in the registers. Then as the position of the raster on

the screen changes, the position of the raster is compared with the register contents. When the position of the raster becomes the same as the eight bits in register 18 and the MS bit in register 17, an interrupt flag is thrown in register 25, the interrupt register.

Interrupt Register

There are five active bits in the VIC interrupt register 25. They are shown in Fig. 20-11. When the state of a bit changes from 0 to 1, it signifies that a flag is thrown and an interrupt takes place.

The interrupts tell VIC that an event has taken place and that the VIC must take appropriate action.

The flag in bit 0 is the one that becomes a 1 when the actual raster position becomes the same as the raster position that has been latched in the raster register. Bit 1 becomes set when the first collision occurs between a sprite and display data. A sprite and display data collision is noted by register 31. Bit 2 is set when the first collision happens between two sprites. A sprite to sprite collision is noted by register 30. Once any of the flags are set, they stay that way till the programmer purposely resets them.

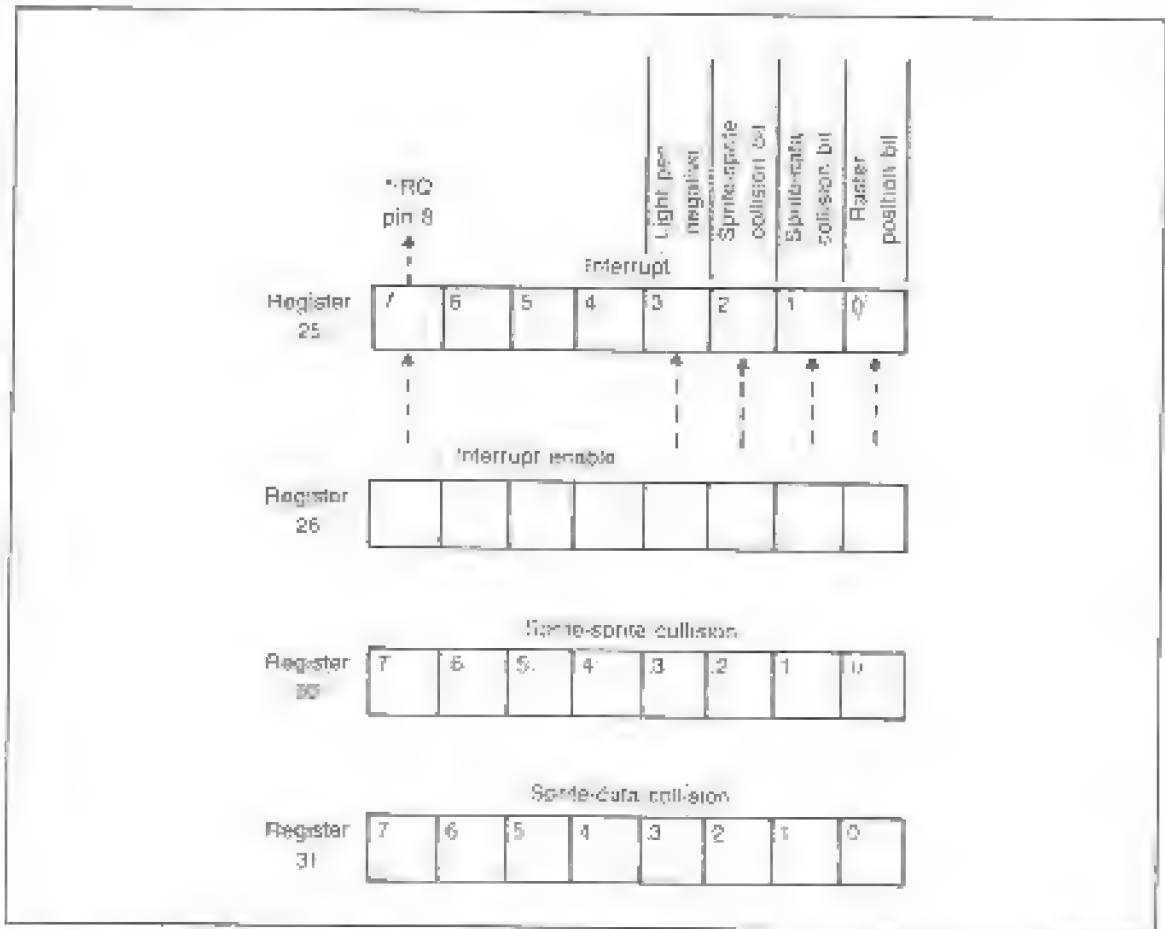


Fig. 20-11 Register 25 is the interrupt. When a collision occurs between a sprite and some display data, bit 1 becomes set. If the collision is between two sprites bit 2 is set. The interrupts are noted by registers 31 for the sprite-data collision and 30 for the sprite-sprite collision. The interrupt can be cleared by writing a 1 in the corresponding bit of register 26 the interrupt enable.

The resetting is accomplished by writing a 1 to the Interrupt Enable register-26. If you write the 1 to the corresponding bit, the same bit in register 25 will clear.

Bit 3 of the Interrupt register is set by the light pen during a negative transition of the input. Bit 7 is set automatically whenever one of the four sources of interrupts happens. This allows the interrupt register to be able to output a low through pin 8, *IRQ of the VIC. Just because bit 7 gets set, that does not mean the low will leave pin 8 and head out into the system interrupt. In order for the low signal to actually leave the VIC, bit 7 of the Interrupt Enable register 26 must also be set to a 1.

The two interrupt registers are very valuable to the programmer. They let him use the screen in a large number of useful ways. He is able to design split screen formats, install eight or more sprites, mix text and graphics and other important graphic techniques. It is important during troubleshooting and repair to have a good idea of the interrupts so that you can make PERK and POKE tests to determine if the registers are operating properly.

LIGHT PEN

Part of the light pen mechanism is inside VIC. The light pen is a device that gives the 64 a touch-pad ability. When the light pen is touched down on the TV screen, the exact position and dot it touches down on is recorded in two registers of VIC. The registers are 19 and 20.

The cathode ray scans the TV screen once every frame. Electrons impinge on every dot once every raster frame. When the electrons arrive at the spot that the light pen is touching, the pen is activated. Since the dot is only hit once every raster frame, the light pen latch is triggered only once every frame.

Register 19 latches the x position of the touch-down. The x position is defined by a 512 bit counter. This means there are 9 bits the counter keeps stepping through from 0 to 511. Register 19 latches the eight MS bits out of the nine. Bit 0 is not recorded.

Register 20 latches the y position of the light pen on the raster. The y position is defined by a 256

bit counter. This is the usual eight bits and the register is able to store them all.

VIDEO OUTPUT

All of the VIC inputs and outputs shown so far have been digital highs and lows. The 49 registers can be read, written to, receive controls, and output controls. All of the processing has been in the digital circuits of the computer.

Internal to VIC are circuits that convert many of the input signals from a digital nature to analog video signals. The VIC is a digital-to-video device. If you test all the signals shown so far, the logic probe reveals their digital states. The same logic probe is useless at the analog output pins.

At the output pins, the ordinary TV service scope makes excellent tests. The output signals are each a portion of a composite color TV signal. At pin 17, the SYNC/LUMINANCE output is a composite signal containing the actual video, the vertical and horizontal sync signals, and the brightness or luminance signal that controls the intensity of the cathode ray. Refer to Fig. 20-12.

Pin 16, the COLOR output, contains all the color information. This includes the chrominance signal, the color burst and all the colors that the display is

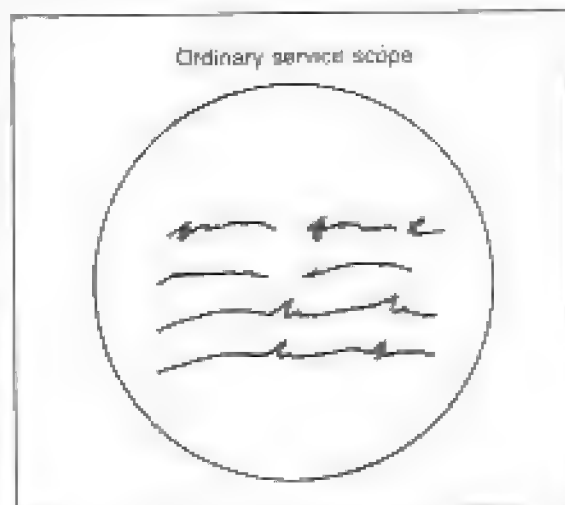


Fig. 20-12. The ordinary service scope can display the TV signals that exit the VIC easily. At pin 16 there should be this TV sync and luminance signal.

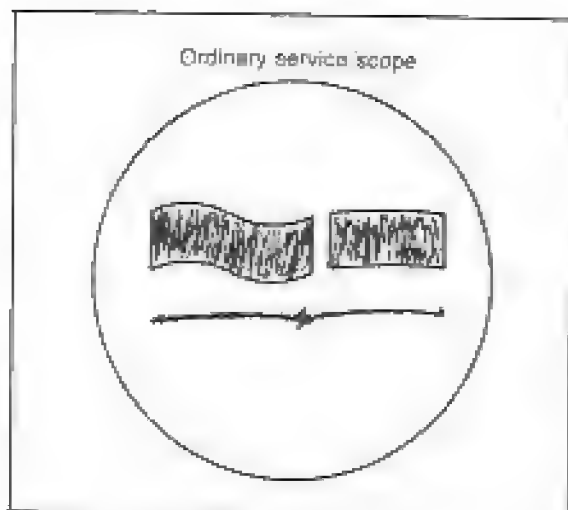


Fig. 20-13. At pin 14, the TV color signal can be viewed on the scope.

supposed to show. Refer to Fig. 20-13. When the two output pins have their signals mixed properly, the result can be input to a CRT for display.

The scope shows the signals well. You can see if the signal is getting out of VIC quickly with the

scope. This is often one of the first test points covered during video type troubles. The resultant composite color TV signal is then attached directly into the rf modulator circuit. It is then installed into a TV Channel 3 rf signal and output to the home TV.

Separate signals of both SYNC/LUMINANCE and COLOR are also connected to the audio-video plug. These signals can be applied to a special TV display monitor that does not require the rf modulation.

Once the signals exit the VIC they are in analog form and must be traced with the ordinary TV service scope. The signals that should be present are all ordinary TV video signals. Any loss of video could be a clue to the trouble you are searching out.

TESTING

The test point chart in Fig. 20-1 can be used to compare your readings from the VIC. Any deviations from the chart indicates trouble. A logic probe or a vom and a service scope are needed to make all of the tests.

21. 80-Column 8563 Video Controller

VIC receives the digital computer work and outputs a composite color TV signal that appears in 1000 character blocks on the display. The display can be sent to the antenna terminals of a TV, or plugs of a color monitor, and it shows 40 columns of characters 25 characters high. The 8563 chip, on the other hand, is quite different. Its output is 2000 character blocks. Its output cannot be sent to the antenna terminals of a TV—it won't display correctly into a 40-column monitor. The 8563 output is special.

The 8563 does receive the digital output from the C128 or CP/M modes, not the C64 mode, and outputs two signals. One is a composite monochrome TV signal that can be inserted into a monitor. The second is an 80-column by 25 row color display called RGBI. The RGBI signal can only be used by a special RGBI monitor like the Commodore 1902 monitor. I have my C128 hooked up to a special 20" Sanyo Model AVM210 monitor TV that has an RGBI input plug.

The 8563 RGBI TV output is excellent for word processing, spreadsheet work and so forth. The

C128 graphics though, are not too easy to manage. It's not the fault of the 8563 chip. The chip has many powerful graphic abilities but the C128 has no commands to use them. If you are an experienced graphics programmer, then you can get some monochrome graphics working. In the future there will probably be information and additional hardware to use the 8563 to the best of its ability.

The 8563 works with its own little group of chips. It has its very own 16K of dynamic RAM on two 4416 chips, U23 and U25, shown earlier in Fig. 18-8. The special address and data bus, DA0-DA7 and DD0-DD7, between the 8563 and its special RAM was discussed in Chapter 18. In its output lines, in Fig. 21-1, the 8563 uses U24, a 74LS244 buffer, to amplify its output for the RGBI port. Also in the output stages are some gates. From U54 there are two OR stages. From U57 there are four buffers and from U29 there are two NOT gates. The output stage also has three transistors: Q1, Q4 and Q5. These components and their duties will be discussed later in this chapter.

Another small 8563 circuit is a crystal oscillator running at 16 MHz. It is the determiner of the width of the pixels in the 80-column display.

THE 8563 RAM

The 8563 is the center of a small circuit envelope. It is an 80-column color video display controller. To do the job, it acts as a form of microprocessor with its own set of 16K RAM. The RAM is located on two 4416 DRAMs, each with a matrix of 16K-by-four bits. Between the two 4416s, they are able to store bytes: U23 stores bits 0-3 and U25 is able to hold bits 4-7. Figure 18-8 shows the connections between the 8563 and its two 4416s. Note that the eight data bus lines, DD7-DD0 from the 8563 to the DRAMs are split in two sections of four lines each. DD3-DD0 go to U23 and DD7-DD4 to U25. The eight address lines DA7-DA0 all go to both RAM chips. The 8563 only needs eight multiplexed address lines to contact the total 16K of the two chips.

The 8563 also generates the *CAS and *RAS to strobe the column and row addresses into the DRAMs and keep them refreshed. The 8563 also connects the read/write line, DR/W to the DRAMs. There are 18 pins on each chip, Fig. 21-2. All but four of the pins, when checked with a logic probe, will read pulses. Pin 4 *WE, on each DRAM is the read/write line and should read high. Pins 9 are connected to +5 volts and accordingly should show a high if okay. Pins 1 and 18 are both grounded and should read low.

The 16K of RAM the 8563 has to itself is used in the following way. First of all, they are used to hold the 80-column video that is to appear on the TV screen. The 80 columns are each 25 characters high. This puts 2000 character blocks on the screen. Accordingly, 2000 bytes are reserved in these RAM chips for video pointers. The first 2000 locations from 0 to 1999 are the 8563's video RAM.

Starting at location 2048 through 4047, a total of another 2000 bytes, the screen attributes are stored. What are screen attributes? They are the color and some other characteristics that appear on the screen. These are the 8563's color RAM and other features. Each character position in the display block is given a one-to-one relationship with the

2000 bytes of video RAM in 0-1999. Therefore each attribute byte gets data that applies to one character block that it is to enhance with attributes. Each of the 2000 attribute bytes contain the bits of information shown in Table 21-1, for the block it will enhance.

Bits 0-3 control the color or intensity of the character in each block. Bit 0 works on the intensity, bit 1 on Blue, bit 2 on Green and bit 3 on Red. Actually, the color can be one of 16, as in Table 21-2. Simply choose the binary bits of one of the 16 decimal numbers and install them in bits 0-3. That color will appear.

The background color for the character block is controlled by a register in the 8563. The register is 26. Bits 0-3 in this register will choose the desired background color for the display. Contacting the 8563 registers are discussed later in this chapter along with contacting the 8563's own RAM.

Bit 4, when set to a 1, will cause the character and background colors to switch back and forth. This appears as blinking of the chosen character.

In register 29 of the 8563 is the underline characteristic controlled by bits 0-4 of register 29. Bit 5 of the attribute byte when set to a 1 turns on the underline feature. A 0 turns it off.

Bit 6 when set to a 1 reverses the character and background colors to give you a Reverse Video. A 0 resets the character and background colors back to their original setting.

Bit 7 when set to a 1 puts 256 more characters into action. These additional characters are the Alternate Character Set.

Locations 8192-16383 in the 4416s are used to store the character set from the character ROM in bank 14. When the CI28 is first turned on, a copy of the character set is loaded into these locations at the top of the 8563's RAM.

When you are in the 80-column mode, as you strike keys, you place pointers into the 2000 bytes in the video RAM. These pointers are accompanied by the screen attributes on a one-to-one basis, to provide the character color, the blinker, the underliner, the reverse video option and the alternate character set. These pointers also tell the 8563 which character in the character set storage area in

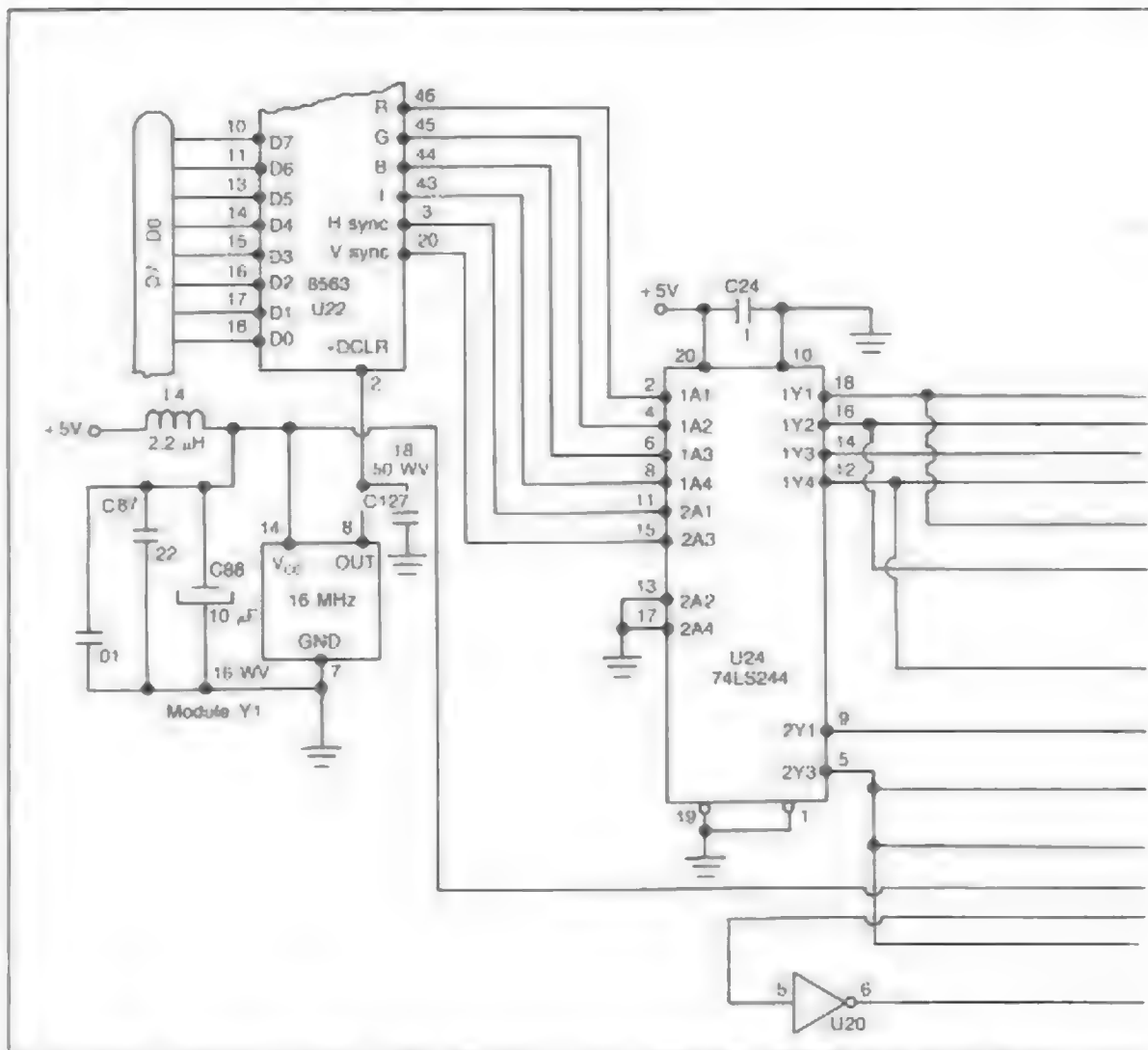


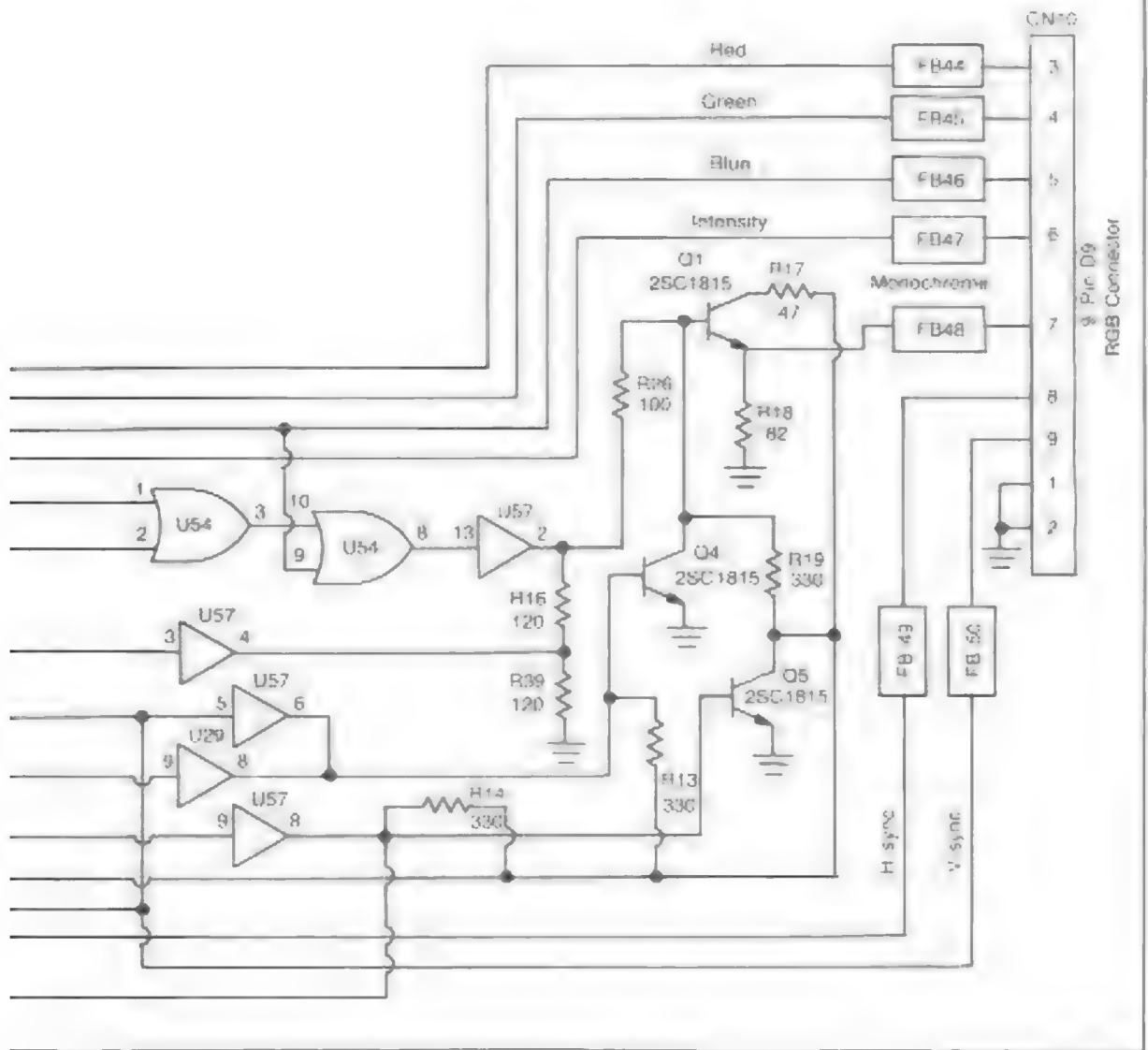
Fig. 21-1 The 8563 uses an amplifier buffer chip, U24, to strengthen and match the RGBI outputs to the subsequent composite monochrome TV circuits and the RGBI port.

8192-16383, to place in the pointer's character block on the TV screen.

The 8563 Registers

Inside the 8563 are 37 control registers. Unlike most of the other registers in the computer,

however, these registers are not on the system memory map. The 8563 system is in an enclave and the registers require special techniques to be contacted. If you are able to get to the registers and their 16K of personal RAM and manipulate them, you can run exercise tests on the 8563 system.



The 8563 only has two locations on the C128 memory map. They are decimal 54784 and 54785. The 8563 is addressed by the processor putting the address bits A11, A10 and A9 into U13, the 74LS138 decoder. The decoder in turn outputs a chip select, CS8563. This turns on the 8563 at pin 7, *CS. The

processor then sends one more address bit to the 8563, namely A0. It goes to pin 8, *RS the register select (Fig. 18-8).

The register at 54784 is the address register for the 37 internal registers. The internal registers are numbered 0-36. The register at 54785 is the data

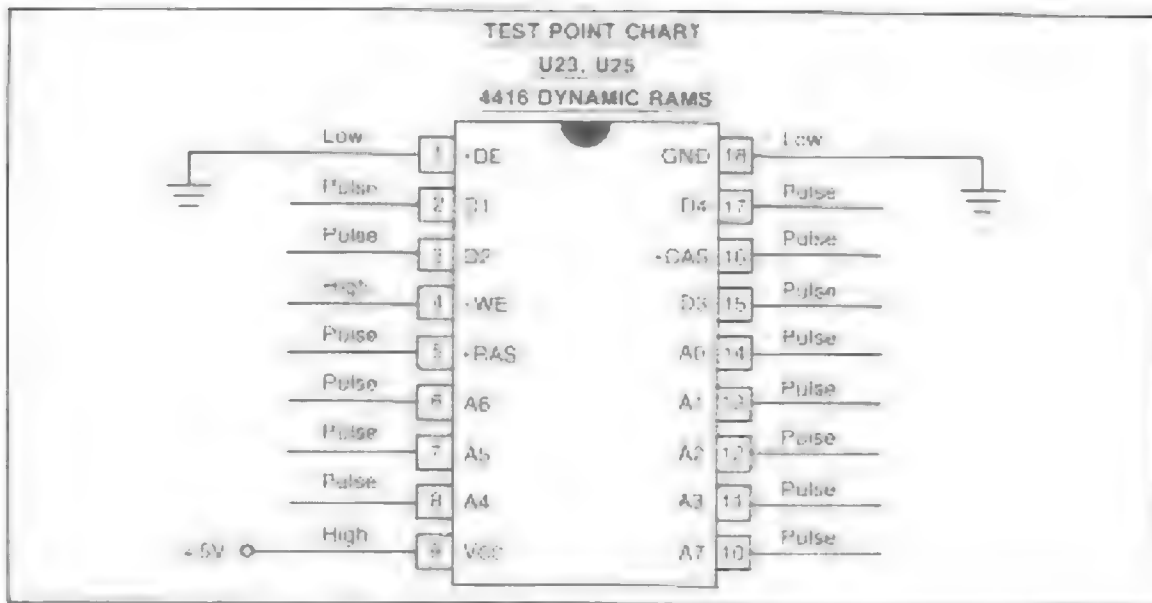


Fig. 21-2 The two 4416 DRAM chips that the 8563 uses as its own private RAM have the same logic states on their pins.

register for the internal registers. These two addresses to the system memory map are the port into the 8563. The 37 internal registers consist of two groups. One is setup registers and the other is display registers. Setup registers are used to make sure the correct characters and attributes get into the proper character block. The display registers produce the correct characters and conduct the attributes on the characters. These registers are vital to make sure that the NTSC standards are in place in America, and PAL standards are in place in Europe.

Table 21-1. The 8563 uses eight attributes to enhance its 2000 character blocks.

Bit Number	Attribute
0	Intensity
1	Blue
2	Green
3	Red
4	Blinking
5	Underlining
6	Reverse Video
7	Alternate Character

In order to read or write to the internal registers or for that matter the 8563's DRAMs, you must pass the bits through the two external registers.

If you want to read the contents of one of the 8563 registers or write bits to the register, then you have to place the register number from 0 to 36 into

Table 21-2. The character block color can be chosen by installing binary bits 0000 to 1111 in bit positions 0-3 of an attribute byte.

80-Column Basic Color Codes	
1	Black
2	White
3	Dark Red
4	Light Cyan
5	Light Purple
6	Dark Green
7	Dark Blue
8	Light Yellow
9	Dark Purple
10	Dark Yellow
11	Light Red
12	Dark Cyan
13	Medium Gray
14	Light Green
15	Light Blue
16	Light Gray

the address register, 54784. Once the register number is in place you can access the register by reading or writing to the data register, 54785. The data register acts as intermediary for all 37 registers.

In order to actually access the 37 registers you will have to write a BASIC or machine language program that performs the required steps. Many such programs are in C128 programming books.

In order to read or write to the 8563's DRAMs, first you must write to the registers as described above. Specifically, register 18 and 19 are the *update RAM location* registers. These two registers are the port into the 4416s. Once you write to them and the write is complete, you can then access RAM. Here again the entire exercise must be done through a carefully designed program found in C128 programming books. A good test program could be one that writes characters to the video RAM in the 4416s. That way, in 80-column mode, the screen would show the characters you write to the addresses 0-1999 in the 8563's personal RAM.

For more details on the registers and what each one does from a programming point of view, please refer to the programming manuals you can purchase to aid you in your programming efforts.

PIN-BY-PIN CHECKOUT

The main symptom of trouble in the 8563 system is, no video on the 80-column display with the 40-column display working well. The prime suspect is U22, the 8563, but trouble could be happening in other parts of the circuits.

When testing video output circuits, good technique dictates starting at the end of the circuits and work toward the beginning. The end of the circuitry is the connection from the buffer, U24, to the RGBI connector. You can start with the logic probe and maybe pick up an incorrect reading.

In U22, the 8563, the final video outputs are exiting pins 46, 45, 44 and 43, Fig. 21-3. They are the bits that turn the pixels that light the screen off and on. They form a 4-bit number that puts one of 16 colors or gray scales into one pixel.

Actually, a color pixel is made up of three phosphor dots, one red, one green and one blue. There

are three cathode beams in the color CRT. Each beam is arranged to light one color phosphor in the pixel. When the pixel lights it emits three color lights but the dots are so small that the lights add and fool your eye into thinking you are seeing one color dot.

A fourth voltage control is also needed, called intensity. The intensity control determines how heavy a beam of electrons will hit the dots. This controls the brilliance of the colors. If a monochrome monitor is used, the intensity controls the gray scale that is produced.

These four pixel lighting signals exit these four pins. Out of pin 46 of the 8563 comes the red video information. It is wired to pin 2 of U24 the buffer, Fig. 21-1. Inside the buffer are a set of amplifiers. They are YES gates and do not invert the signal. A low enters pin 2, passes through a buffer and then exits pin 18. A low should be on pin 18 too. The line goes to the R pin of the RGBI port.

The identical signal movement happens for the green signal coming out of pin 45, the blue signal from pin 44 and the intensity signal from pin 43. You can quickly check out the 8563 output, the U24 input and output and even the connection at the RGBI port. If a reading is wrong you have a clue as to where the signal has met foul play.

Coming out of pins 20 and 3 are the vertical sync and horizontal sync signals to go with the RGBI video. The horizontal sync locks the horizontal oscillator in the TV monitor in step with the desires of the 8563. The vertical sync signal locks the vertical oscillator in sync with the vertical output of the computer.

Pin 20 of the 8563 outputs the vertical sync signal. The signal enters the buffer at pin 11, is amplified and is then sent to the RGBI connector from pin 9. Pin 3 outputs the horizontal sync signal from the 8563. It enters the buffer at pin 15, exits at pin 5. All of the sync lines will show pulses.

At the two sync pins, the signal is a TV type and can easily be seen on the ordinary TV service scope. Figure 21-4 shows what the pulses should look like on the inexpensive scope. Pin 20 of the 8563 shows the vertical pulse and pin 3 the horizontal sync.

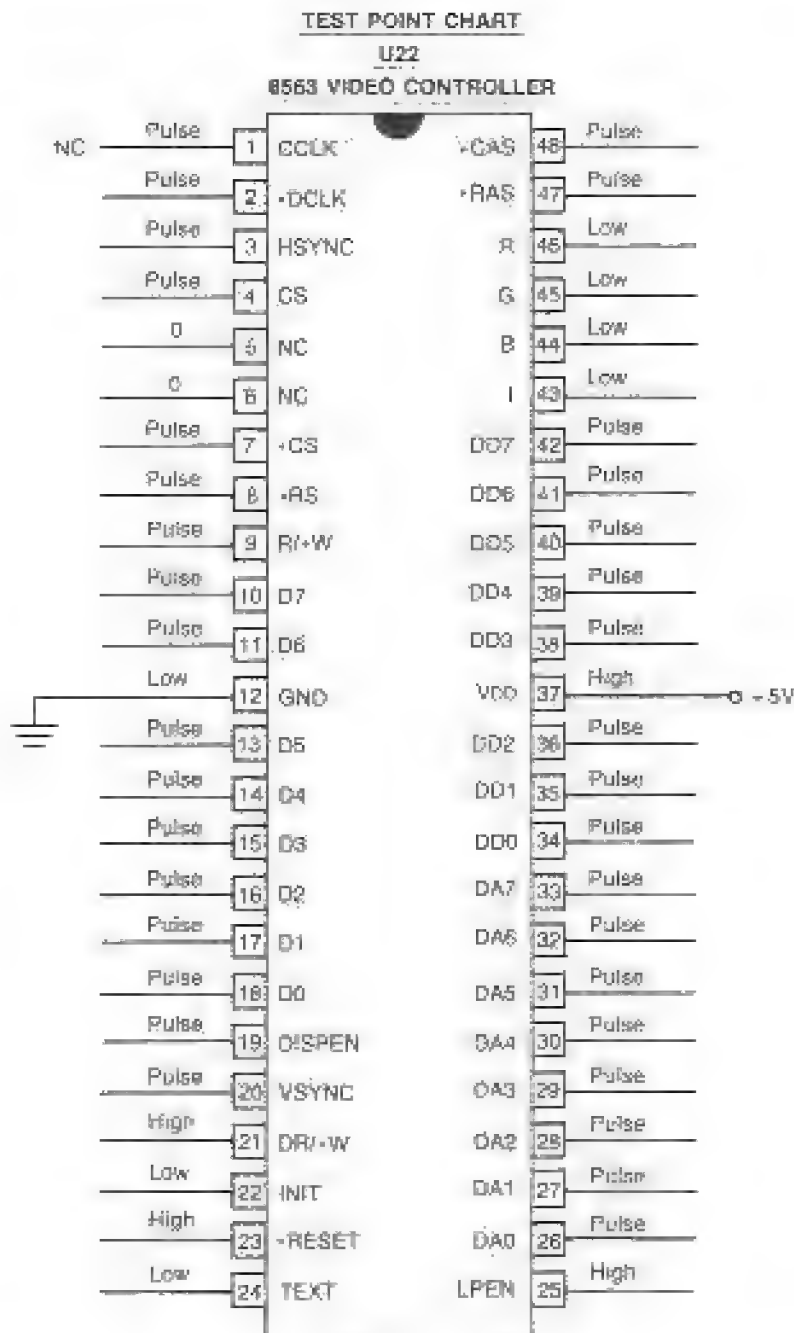


Fig. P1-3. The 48-pin 8563 has logic states on all pins except 5 and 6 that are not connected.

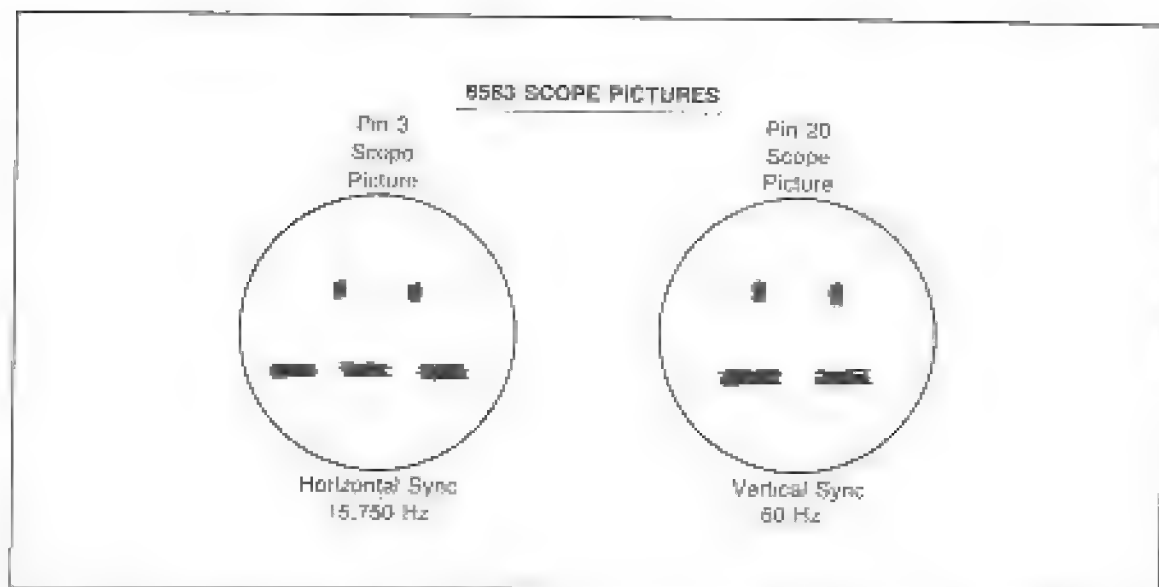


Fig. 21-4 Pin 3 and 20 output the horizontal and vertical sync pulses for the 80-column display. An ordinary TV repair scope will show the sync signals clearly.

The Monochrome Composite TV Signal

In the output lines from the 74LS244 buffer to the RGBI port in Fig. 21-1, note the taps on each of the six output lines. The taps from the red and green lines go directly to an OR gate, part of U54. The tap from the blue line goes to another OR gate in series with the first one. All three lines are thus joined together, and put through a buffer from U57. The intensity line tap is simply passed through another buffer from U57. The two buffers, one containing R, G and B and the other containing I, have their outputs joined. The resultant voltages are then injected into the base of Q1, a video amplifier transistor.

Meanwhile, the two taps from the vertical and horizontal sync outputs are connected to more buffers. The vertical sync is tied to two gates, one NOT gate from U29 and one YES gate from U57. The horizontal sync is connected to one YES gate from U57 and a NOT gate from U29. The sync outputs are then amplified in Q5 and Q4. The outputs are joined and then joined again with the RGBI output in the base of Q1, the final video amp. All this sync and video mixing processed a workable monochrome composite TV signal. Q1 performs the final

amplification and has its output wired to pin 7 of the RGBI connector. At this junction there is an 80-column monochrome composite TV display signal.

There has been some confusion among people about this signal. It is a separate signal from the R, G, B and I outputs. It has nothing to do with them except for the fact that it was constructed from the six U24 output signals as were the R, G, B and I signals. The confusion is that a lot of people selling and using the C128 are not aware of this separate monochrome signal. It does exist and works fine on a monochrome monitor.

The Video Dot Clock

At pin 2 of the 8563, the logic probe will read a pulse. You need an expensive scope to see this sine wave signal because it is 16 MHz—even higher than the C128 master frequency which is 14.31818 MHz. Pin two is connected to it, a 16 MHz crystal oscillator module with accompanying capacitors. This is the Video Dot Clock, called DCLK. This is really the master frequency for the 8563 circuit enclave. The 8563 makes its own master frequency to deal with its own personal circuits.

The higher frequency, first of all, is needed to size the pixels for RGBI display purposes. The 40-column system uses a wider pixel than the 80-column. A wider pixel turns off and on at a slower rate. In 80-column work, the narrower pixel must be working with a faster frequency so it can turn off and on faster.

The 16 MHz is also used by the 8563 to develop the timing for the vertical and horizontal sync pulses. The master frequency is divided inside the 8563 to produce the desired frequencies that emerge out of pins 20 and 3. The 16 MHz signal performs other chores too. One is not used in the C128. It is called the character clock, CCLK, and uses pin 1. Even though the CCLK is unused, if you probe pin 1 you'll find a pulse is present, Fig. 21-3. This pulse is a result of internal wiring. When the 8563 is running, however, that pulse, for testing purposes should be present. If it isn't there then the chip could be defective.

The DCLK is also used by the 8563 to time out its 16K DRAM. It generates the pulses you will find on the internal address and data bus lines: DA0-DA7 at pins 26-33; DD0-DD7 at pins 35, 36 and 36-42. It produces the pulses on the *RAS and *CAS lines at pins 47 and 48.

The Read/Write Lines

The 8563 has two read/write lines. One is an input from the 8502. It connects to pin 9 and is called FR/*W. The actual R/*W line leaves the 8502 at pin 39. The line is held high by a 4.7K resistor at +5 volts. The R/*W line then connects to an AND gate from U8. The output of the AND gate is FR/*W that connects to pin 9 of the 8563. Pin 9 will probe a pulse.

The FR/*W line enters the 8563 and controls the direction of the system data bus, D7-D0, that is connected to pins 10-18, except for 12 which is chip ground. All the pins probe out pulses except for 12 at ground which is a low. This data bus is used to access the 37 registers by the 8502.

Out of pin 21 comes the second read/write line: DR/*W. This line is produced by the 8563 to control the direction of its personal data bus, D17-D10,

out of pins 34-42, except for 37 which connects to +5 volts. All of the data bus lines probe pulses. Pin 37 though will probe a high because it is delivering +5 volts to the chip.

DR/*W comes out of pin 21 of the 8563 and connects to pins 4 of the two 4416s. The DR/*W line does not probe a pulse. It will read a high on the logic probe. Pins 4 of the 4416s will also read a high because they are direct connections to the DR/*W.

The Other Control Lines

Two chip select pins are on the 8563. One is CS at pin 4. Note there is no asterisk in front of the CS, which means it must be high in order for the chip to be selected. The signal entering pin 4 is the 2 MHz pulse from VIC. This signal ties the 8563 into sync with the rest of the system. The chip is selected at pin 4 everytime the 2 MHz square wave goes high.

The other chip select at pin 7, *CS, has an asterisk. This pin is made active when the signal is low. The signal that enters this pin is the *CS8563 that was created from system address lines and arrived from the decoder U3. When CS is high and *CS8563 is low, then the 8563 chip is turned on. Both of these pins will probe out as pulses when operating correctly.

Pin 8 probes a pulse too. It is the register select, *RS. The pin is tied to address line A0. A0 can be a high or low according to the address the 8502 puts out from its program counter. When A0 is a high, the 8563 will permit reads and writes to its input data register. These reads and writes are then conducted between the processor and the 37 internal registers of the 8563.

When A0 is low, the 8502 can write to the address register in the 8563 to pick out the one out of 37 internal registers it wants to choose. A low also permits a read of the status register.

Between the two chip selects and the action of address line A0 into the 8563, the two input registers can be activated so that the 8502 can get to the 37 internal registers.

At pin 22 of the 8563 is the INT (Initialization) input. It is tied to ground along with pins 24 and 12.

Pin 12 is the 8563 normal ground and pins 22 and 24 are tied there to keep them out of the way. They will all probe a low since they are at ground.

The reset action in the 8563 is conducted through pin 23, $\overline{\text{RESET}}$. This pin will probe a high because it is tied to the system $\overline{\text{RESET}}$ line. When the line is low, as during initial start up, the line causes the 8563 internal latches to be cleared. This is the initialization the 8563 needs to begin operation.

Some pins on the 8563 are completely unused. They are:

- Pin 19, DISPEN , display enable, no connection, probes a pulse.
- Pins 5 and 6, no connections, no probe reading.
- Pin 7, the character clock, CCLK . It probes a pulse from internal wiring to it.

At pin 25 is the input for the light pen when used. It probes a high. When it is a high it will place into latches the vertical and horizontal block positions of the character it is dealing with on the display.

22. Sound Interface Device

The Sound Interface Device is the audio companion to the VIC and the 8563. The SID is designed to take bytes or bits from the processor, convert the bits into audio signals, and send them to the speaker in the TV monitor or a separate audio system. The created audio produces all the sound effects to further excite the graphics or to imitate musical instruments.

The SID has 29 addressable registers that create the sound effects. You can write to 25 of the registers with the BASIC POKE statement. The registers that can be written to cannot be read from. You can read from the remaining four registers with the PEEK function. The read registers are also one directional. They can only be read and not written to.

PINOUT

The 8502 sees the SID as 29 addresses on the memory map. When the processor wants to access the SID it outputs a register address. The higher address bits are then sent over the address bus and

arrive at the PIA chip. The bits are decoded and an I/O pulse continues on to the U3, a 74LS138 as in Fig. 22-1. The output is connected to the *CS pin 8 of the SID. When the SID is dialed up, a low enables pin 8. The pinout is shown in Fig. 22-2.

That one low though will not fully address the selected register. The SID needs more signals. It needs a clock driving pulse to open up the register. The pulse is 1 MHz and enters the SID at pin 6. During the high SID can be accessed just as RAM or ROM is. Both the reads and the writes are performed during the high of 1 MHz.

The SID knows to receive data or output data according to the input on pin 7, R/*W. If R/*W is high SID allows the processor to read the register. When R/*W is low SID accepts data bits from the 8502.

The registers themselves are addressed with the five lowest address lines, A4-A0. With five lines, a total of 32 registers could be addressed. Since the SID only has 29 registers, that leaves three extrane-

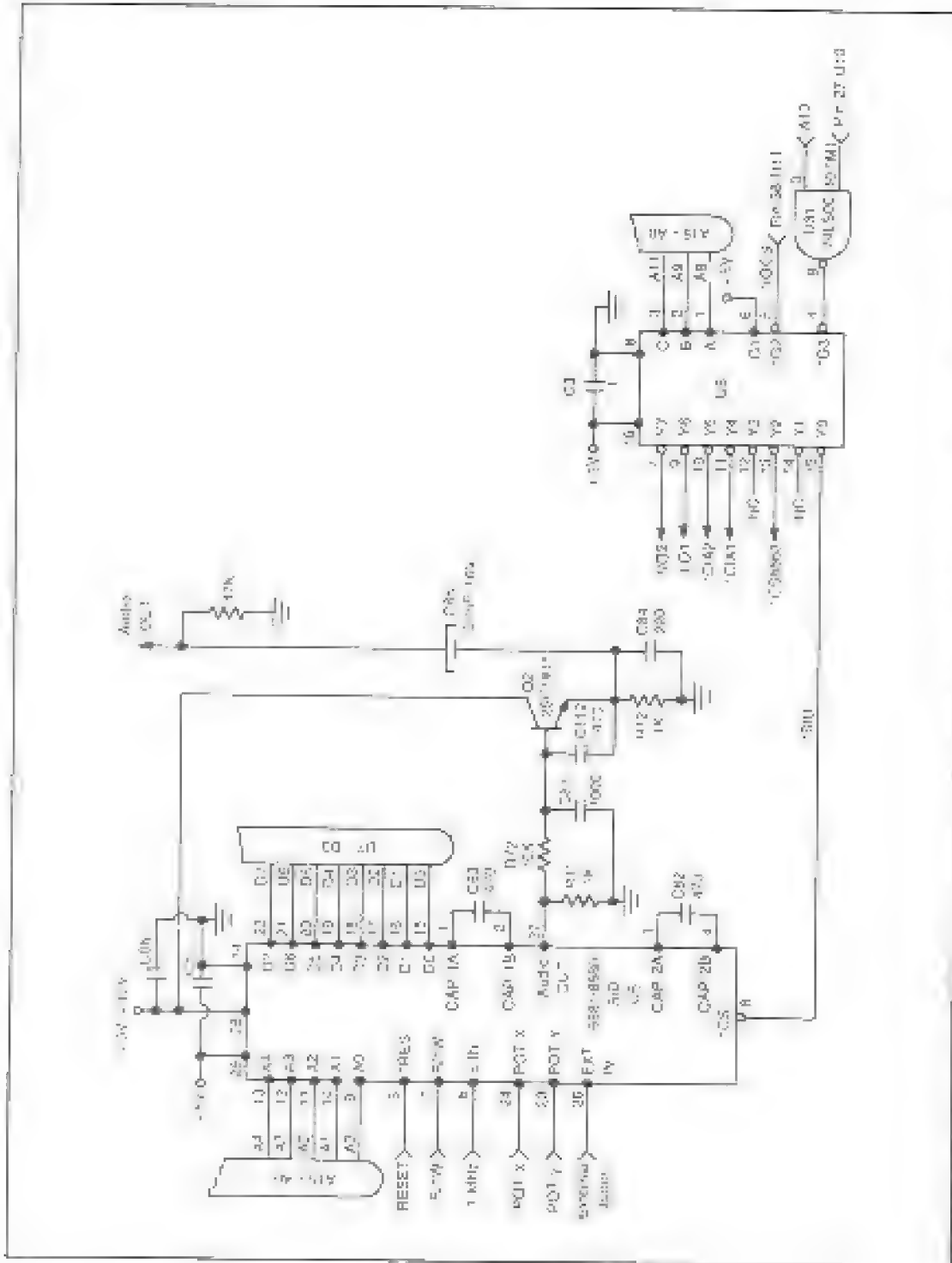


Fig. 22-1. SID is selected by a -SID signal from the U3 decoder. It then works directly with the processor through the system address and data bus lines

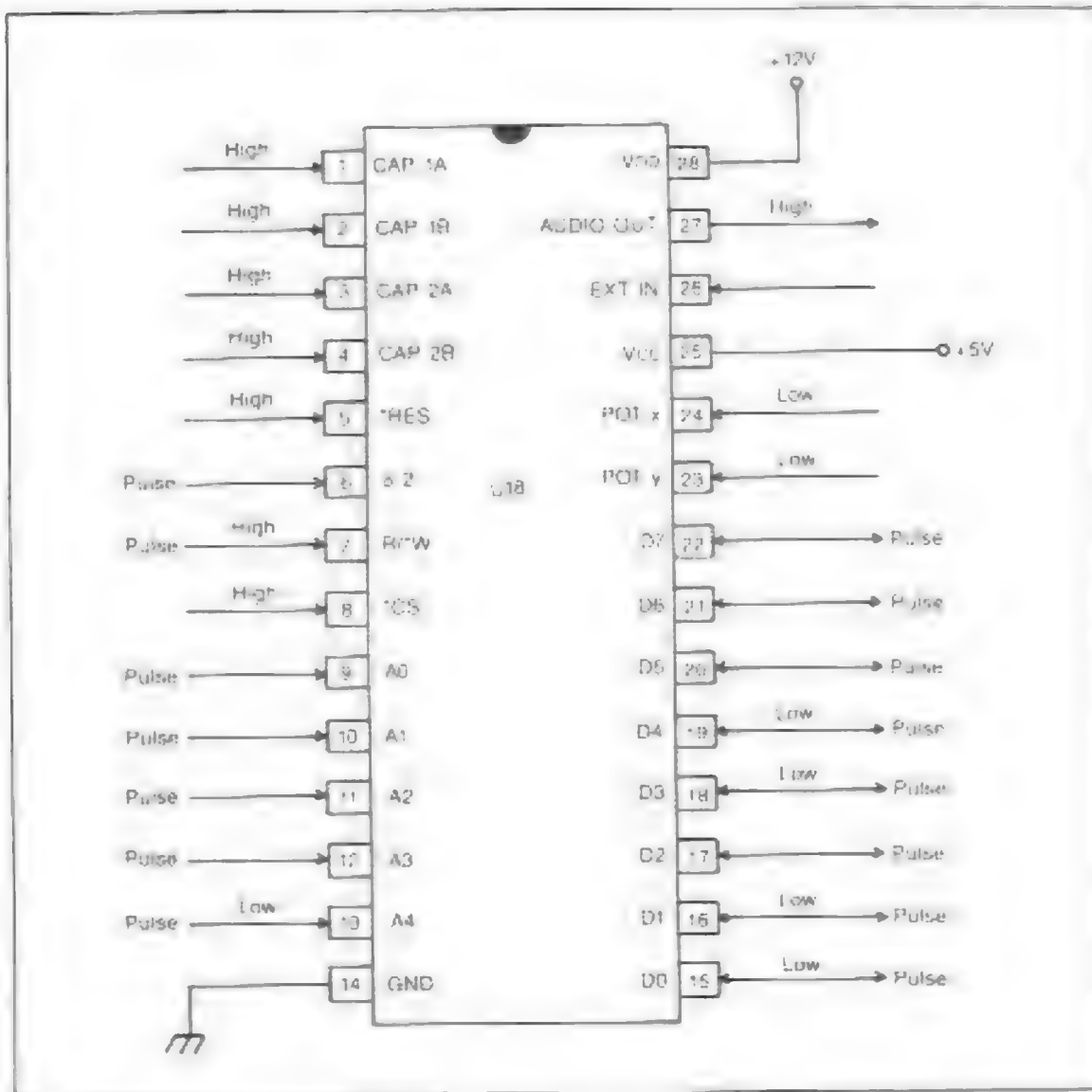


Fig. 22-2 The SID is a complex chip that needs two different supplies. There is +12 volts at pin 28 and +5 volts at pin 25.

ous addresses. Should one of the extra registers be written to, SID simply ignores the data that arrives from the processor. If one of these registers is mistakenly read, data will arrive at the processor but it will be meaningless.

The data lines to the SID are the normal D7-D0 attachments in the system. They connect to pins 15-22 on the SID. There are data buffers in the SID

to handle the data transfer. The three-state buffers are on when the 8502 writes data to the SID, or accepts data when the SID is read.

Pin 5 on SID is the *RES input. If the reset pin is brought low for at least ten cycles, the SID will reset all of its registers to zero. This stops any audio output that might be going on. The reset pulse is the same one that originates in the 556 and starts

the processor off. The SID is started at the same time.

Pin 14 is ground. It is a good idea to always reconnect SID's ground to the power supply and not anywhere near the other digital circuits. The VIC should be connected separately in the same way. Both the audio and the video outputs can be damaged by noise from the fast moving digital bits that are traversing the digital bus lines and chips.

Pin 28, VDD, is the +12 volt supply that energizes the drains of the FETs in the SID. VDD receives a separate power line and is filtered and bypassed thoroughly. The capacitor removes high frequency noise, medium frequency noise, and low hum type noise that might appear.

Pin 25, VCC, is the +5 volt supply that energizes the collectors of the transistors in SID. This line too is separate in the effort to keep noise levels as close to non-existent as possible.

Pin 27 is the audio output pin. It exits from an open source buffer and contains all the outputs from all the audio generators in SID.

The total audio output energy is set by an internal volume control. The volume is controlled by the programmer and the data he sends to the control. It has a peak-to-peak value of 2 volts at a dc level of six volts. A 1K resistor returns the source to ground.

The output signal in this machine is coupled to the base of an npn emitter follower in Fig. 22-1. The audio is removed from the npn through a 25 working volt, 10 μ F series filter. The 6 volt dc level is ac coupled in this manner. The signal is output to the rf modulator and also can be tapped off at the subvideo plug.

SPECIAL INPUTS

The SID has some other very useful inputs. First of all there is pin 26. It will accept audio signals from an external source such as your voice or a musical instrument. Any signal that is input here should enter at a dc level of 6 volts and be larger than 3 volts peak-to-peak. This is accomplished in the same way the SID outputs audio. A series 10 μ F filter will perform the input chore. The input impedance at this pin is about 100K ohms.

The signal that is input at pin 26 is mixed with the audio that the SID is putting out. It is also passed through the filter. This input is valuable in case you want to use a number of additional SID chips. The only restriction to the number of additional chips you can input is the noise that each additional chip adds to the output. A large number of chips can be input. The volume control in the SID is positioned to act on the total input as well as SID's own output.

Two more special inputs are at pins 23 and 24. Refer to Fig. 22-3. They are called POT x and POT y. These inputs are connected over a pair of 1000 pF bypass capacitors. The two capacitors are designed to operate with a pair of external 470 ohm potentiometers. The pots connect to +5 volts and varies the input voltage.

The pins are inputs to the analog to digital converters inside the SID. The A/D converters take the analog dc voltage and convert it to a digital set of highs and lows. The capacitor is designed at 1000 pF for the SID to keep pot jitter at a minimum.

The last special inputs are at pins 1, 2, 3 and 4. These hold the integrating capacitors for the filter. All of these inputs are discussed in more detail later in this chapter. The SID has two of these capacitors installed. Each has a value of 470 pF. One is installed between pin 1 and 2, while the other connects 3 and 4. They are fairly well matched. The capacitors are the polystyrene type.

The filter operates over the audio range between 30 Hz and 12 kHz. This is about the range the normal home TV audio outputs. In special audio applications, you could change the capacitors to different values to extend or reduce the frequency response.

OPERATION

The SID has three sets of circuits that are all quite alike. Each circuit set is designed to output an individual *voice*. The choir operates as a total voice or each circuit can sing a solo.

Each circuit set contains a tone oscillator, a waveform generator, an envelope generator and an amplitude modulator. Refer to Fig. 22-4. The tone oscillator is variable and can be tuned to produce different frequencies. This is called the pitch of the

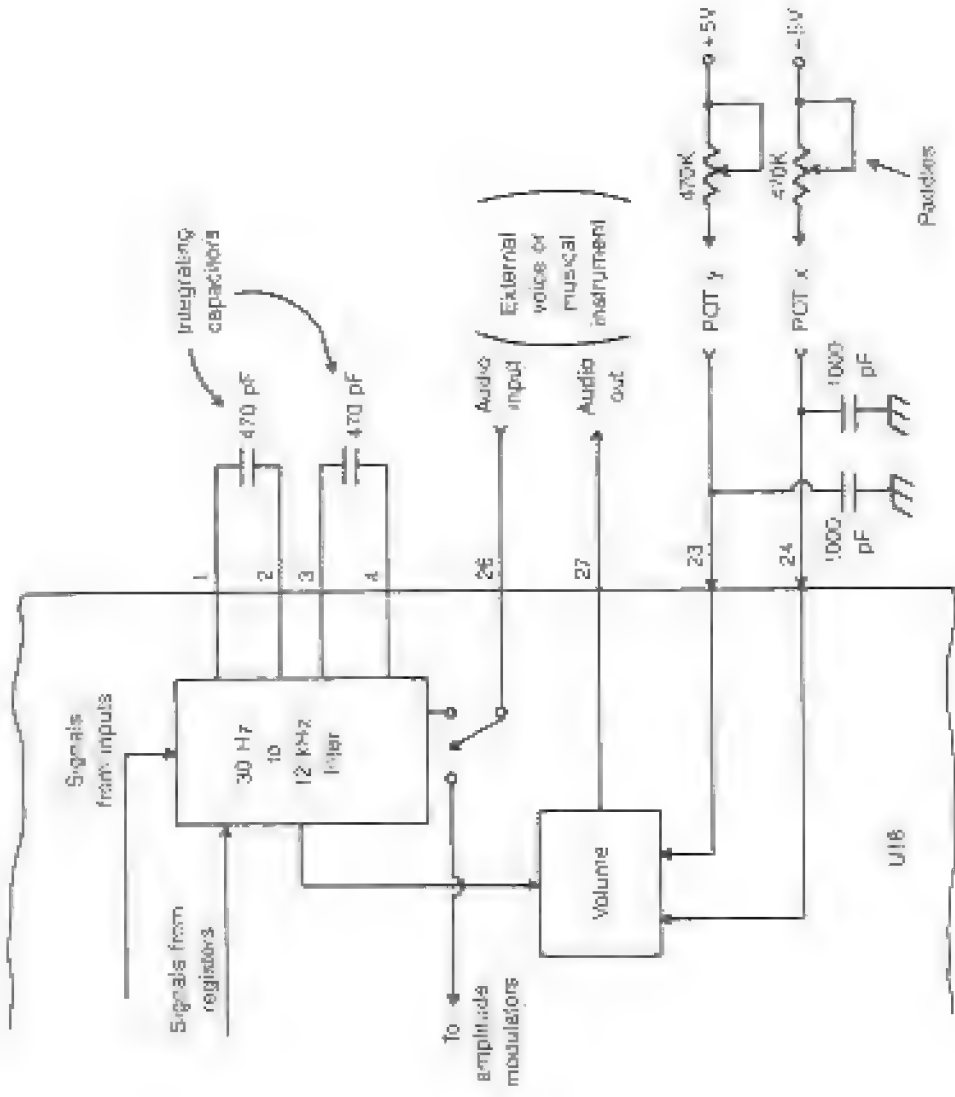


Fig. 22-3. Besides the audio input pin at 26, the SID is able to receive paddlin inputs at pins 23 and 24.

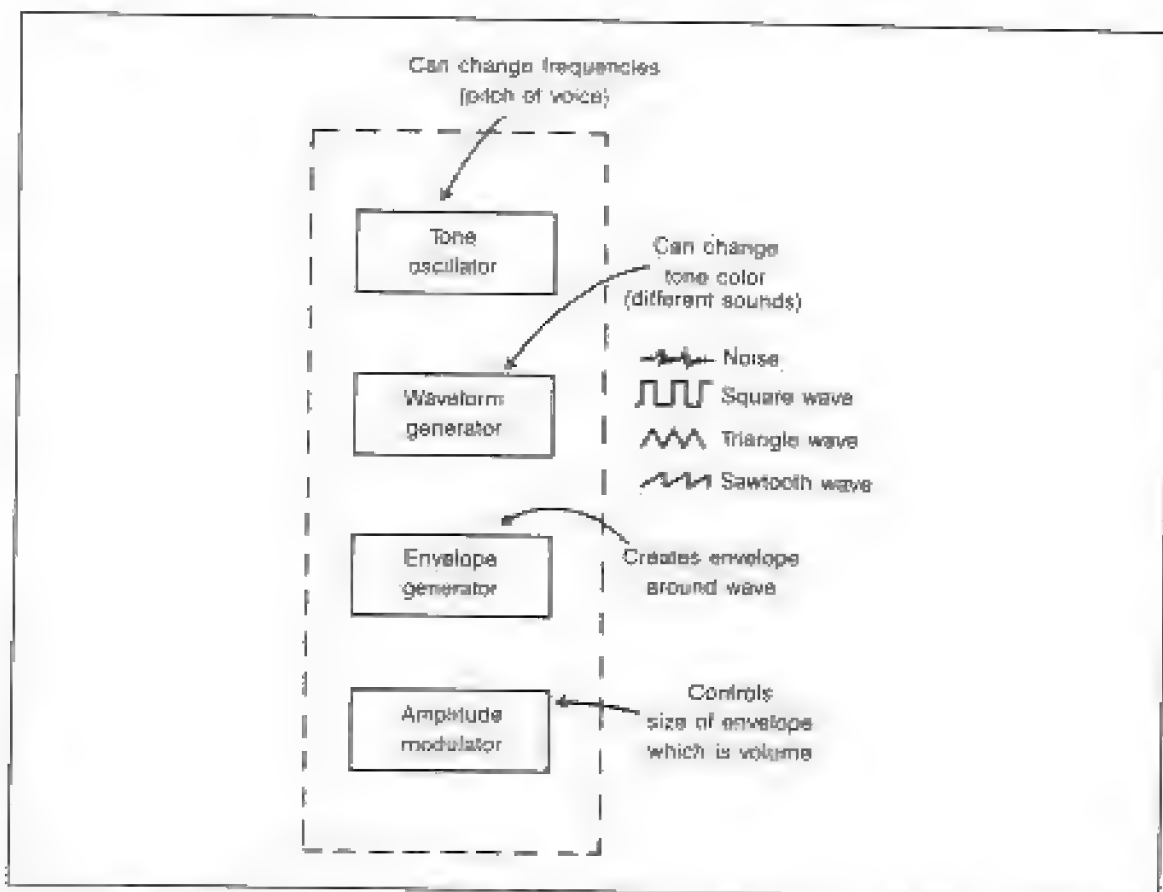


Fig. 22-4 Each voice is produced by means of four circuits working together. There is a tone oscillator, a waveform generator, an envelope generator, and an amplitude modulator.

voice. The pitch can be varied over the range the TV audio will reproduce.

The waveform generator is able to take the oscillator frequency and produce four different waveshapes at that frequency. The different waveshapes of the same frequency cause variations in the audio. The differences in sound are called tone color.

The loudness of the audio is controlled by the amplitude modulator. The envelope generator is able to create an envelope around the audio to control the volume. The volume is programmable due to the ability of the envelope generator.

The 25 write-only registers produce the sound effects, on order, as they receive data from the

8502. The processor is also able to read the four read-only registers. Two of the four registers contain the values of POT x and POT y; Another of the registers has data from the envelope register in the third set of circuits. The last register contains the constantly changing output of the third tone oscillator.

These read outputs from the SID can be used in a number of ways. They can be mixed in with other audio to produce sound effects. The changing frequency of the third oscillator has a random movement. It is used to generate random numbers that can be used as the basis for some games. Let us examine the activity in the 29 registers as they operate.

REGISTERS

The S1D has five sets of registers. They are shown in Fig. 22-5. There are three sets of voice registers, one set of registers for the filter and a set of four registers to handle the odd jobs. The

voice and the filter registers are all write-only. The four odd job registers are the read-only types.

Each register is addressed internally by means of five bits that enter through address lines A4-A0. The registers are all byte size. The bits in the

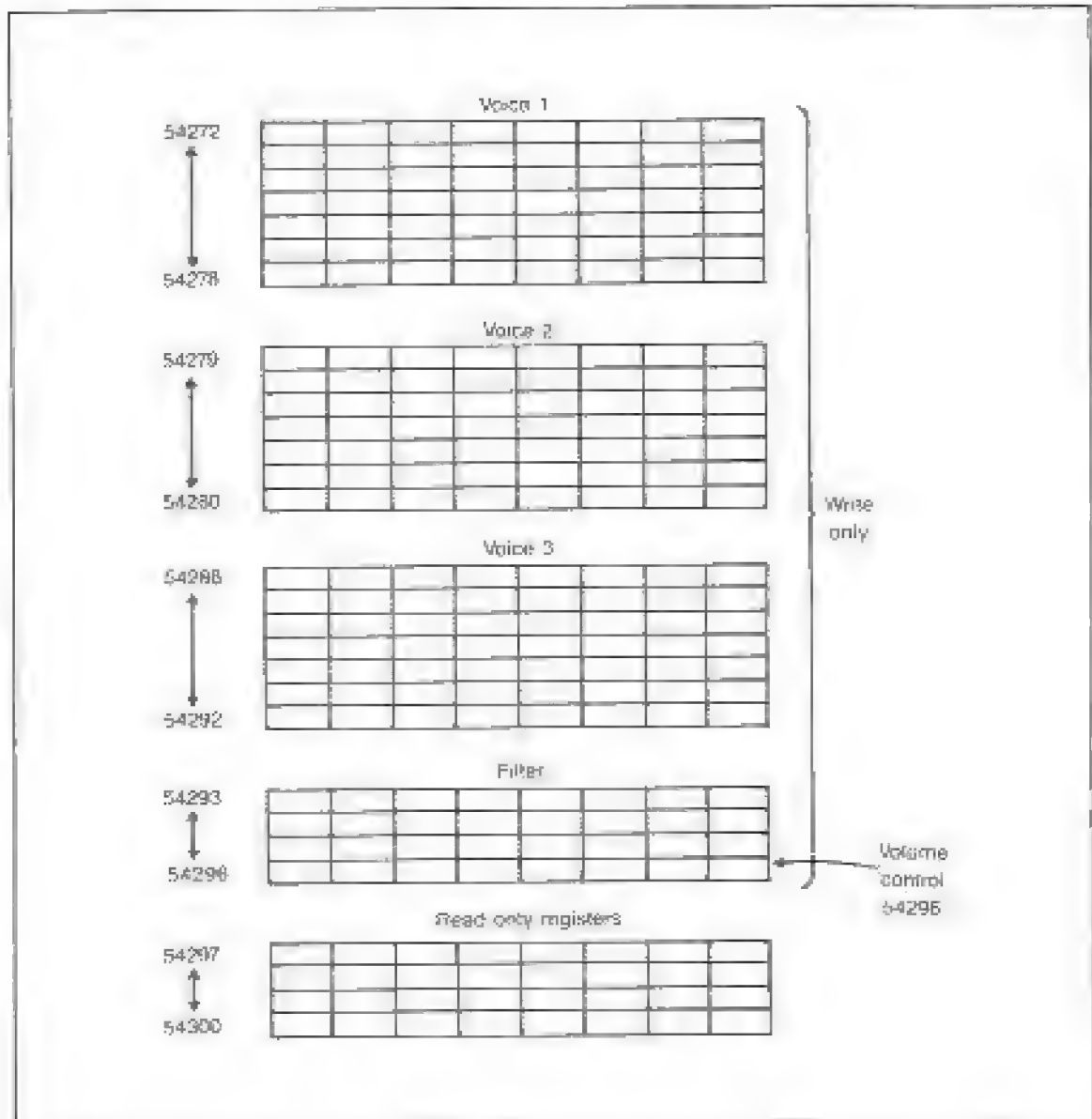


Fig. 22-5. The S1D contains three sets of voice registers, one set of registers for the filter and one set for ready only uses.

registers control the circuits that generate the audio output. Each bit is like a push button on a control board to produce sound effects and other related functions. There are seven registers assigned to each voice.

First Voice Registers

Of the seven registers in Fig. 22-6, two are called Frequency High and Frequency Low. They are wired together and form a 16-bit register. The register controls the frequency of the tone oscillator in the first voice. If you POKE a decimal number into the 16 bits, the SID will generate a musical note. In your users guide, there is a table of the numbers and what musical notes they will generate when POKEd.

Two more registers are called Pulse Width High and Pulse Width Low. These registers are also wired together to form a two byte register. However, only 12 of the 16 bits are used. Four of the higher Pulse Width High bits are unused. This 12-bit register controls the pulse width of the waveform that comes out of the tone oscillator. This allows the audio output changes to be smooth as the tone changes from frequency to frequency.

The pulse can vary from a constant dc output at one extreme to a square wave with equal highs and lows at the other extreme. To produce a constant low, 0000 0000 0000 is installed. To produce a constant dc high, 0000 1111 1111 is placed in the register. If you want a square wave, 0100 0000 0000 is installed. For pulses inbetween, other binary bits are placed in the registers.

Voice Control Register

The eight bits in this register make the tone oscillator perform in different ways. Bit 0 is called the Gate bit. The actual details are discussed later in the chapter but the bit controls two functions. When the bit is set to a 1, that triggers the envelope generator and the ATTACK/DECAY/SUSTAIN cycle is begun. If the bit is cleared the RELEASE cycle takes over.

Bit 1 is the SYNC. When it is set to 1, tone oscillator 1 is synchronized with tone oscillator 3. This aids in producing desirable harmonic sounds.

Bit 3 is the TEST bit. When it is set to a 1 it resets and locks the tone oscillator at zero. It also resets the noise waveform of the oscillator and the pulse waveform output is placed at a steady dc level. This positioning is needed for program tests. It also is useful for the programmer to be able to synchronize the oscillator with sound that is occurring outside SID.

Bit 4 when set to 1 is the control that turns the triangle waveform on in the tone oscillator. The triangle waveform is one of the ways the oscillator can run. As an audio output, it sounds something like a flute.

Bit 5 when set to 1 turns the sawtooth waveform on in the oscillator. It has sort of a brass instrument effect.

Bit 6 when set to a 1 turns on the pulse waveform in the oscillator. The Pulse Width can be adjusted by writing to the PW registers as discussed before. All types of interesting sounds can be produced from a hollow booming square wave to a squeaky reedy peep.

Bit 7 is the Noise output. Noise by its very nature is a true random device. The random noise can be adjusted from rumbling to hissing or whatever by the tone oscillator when bit 7 is set to a 1. This is the bit that can produce the explosions, rocket motors, windstorms, waves, drums, cymbals, and so on.

Envelope Generator Registers

The ATTACK/DECAY register and the SUSTAIN/RELEASE register control the envelope generator circuits. When a sound begins and rises in volume, that is called the *attack* rate of volume. When the sound then peaks out and falls in volume, that is called the *decay*. There is an 8-bit register in the voice that controls the envelope generator that produces the ATTACK/DECAY. The four highest bits are attack bits. The four lowest bits are the decay bits.

Voice 1	7	6	5	4	3	2	1	0	Frequency low
54272									
54273	15	14	10	12	11	10	9	8	Frequency high
54274	7	6	5	4	3	2	1	0	Pulse width low
54275	—	—	—	—	11	10	9	8	Pulse width high
54276	7	6	5	4	3	2	1	0	Voice control
54277	3	2	1	0	3	2	1	0	Attack-decay
54278	3	2	1	0	3	2	1	0	Sustain-release
Voice 2									
54279									

Fig. 22-6. In each voice, there are seven control registers

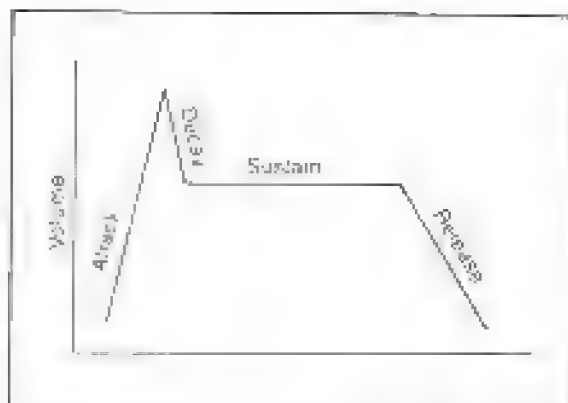


Fig. 22-7 When a sound begins and increases in volume, it is called the attack. As it peaks out and falls in volume it is called the decay. The mid range of sound after attack and decay is known as sustain. When the sound quits, it goes into the release.

The mid range of a sound that goes through attack and decay is called the sustain. *Sustain* is a volume level. When the sound quits, it falls to no volume at all, dead quiet. The rate at which the volume falls to nothing is called *Release*. Figure 22-7 illustrates these four concepts.

A companion register to ATTACK/DECAY is the SUSTAIN/RELEASE register. For more details on how to use the registers during programming see the user manual that came with your C128. The bits in the SUSTAIN/RELEASE register are laid out like the ATTACK/DECAY register. The higher bits are used for the sustain control and the lower bits for the decay. When the programmer uses these registers, he must follow carefully all the rules of operation. Some truly amazing sound effects can be produced with attention to the details.

The Other Voices

The two other voices and their seven registers are essentially identical to the first voice. The only difference between them are the synchronization of the oscillators and the ring modulated replacement of the triangle waveform.

When the voices are operated, you would have to select first of all the frequency you want to produce. Next there is the choice of the waveshape. Third you must pick out the effect that is needed.

This is done with the SYNC and RING MOD bits. The envelope rates are decided on after that. When to turn the sound on and off and the length of time you want the sound to stay on is figured next. The voices can be used as solos, or as a choir.

Filter Registers

The first two registers, FC LOW/FC HIGH in the filter set, are combined to form a double-byte register. However, only 11 bits are needed for the job so bits 3-7 of FC LOW are not used. The job the registers perform is to set the cutoff frequency of the filter. The frequency used is 30 Hz to 12 kHz. The registers can be written to, which makes the filter a programmable circuit. Refer to Fig. 22-8.

The filter can be tuned to resonate at particular frequencies. When it is tuned to a frequency, sounds at that range have a much sharper sound. All the frequency components in that range are emphasized.

The higher bits of the RES/FILT register are wired so they can tune to a particular frequency range. Since there are four bits to the control, 16 resonant settings can be made. 0000 produces no resonant effect, and 1111 produces the finest tuning.

Bits 0-3 of the register serve a different purpose. Each bit acts as a switch to route four different signals through the filter. Bit 0 works on the route the first voice takes to the audio output pin 27. When bit 0 is cleared to 0, the voice bypasses the filter and goes directly to the audio output. When bit 0 is set to a 1, the voice is forced through the filter and is processed accordingly.

Bits 1 and 2 perform the exact same routing job for the second and third voice outputs. Bit 3 also routes a signal the same way. It handles the audio input that enters pin 26. A 0 in the bit bypasses the audio input around the filter. A 1 in the bit forces the audio input through the filter.

Mode/Volume Register

Here again the eight bits in the register are separated into two nybbles, with the nybbles performing similar jobs with its four bits. The higher nybble, bits 4-7, decides on the mode of operation the filter will perform in. Bit 4 is called the LP

Filter					2	1	0		
54293	—	—	—	—					FC low
54294	10	9	8	7	6	5	4	3	FC high
54295	3	2 (Resonance)	1	0	External in filter	3 Filter 3	2 Filter 2	1 Filter 1	Resonance filter
54296	Voice 3 off-on switch	High pass	Band pass	Low pass	Volume 3	Volume 2	Volume 1	Volume 0	Mode volume

Fig. 22-8. The filter sets the cutoff range of the audio between 30 Hz to 12 kHz.

bit, since it works with the low-pass qualities of the filter. When bit 4 is set to a 1, the low-pass output of the filter is used. All frequencies below 30 Hz are passed as is. The frequencies above 30 Hz are attenuated. This produces hearty sounds.

Bit 5 is called BP for bandpass outputs. The passband for the audio output is between 30 Hz and 12 kHz. When bit 5 is set to a 1, the frequencies in the passband are allowed through and the frequencies above and below the band are attenuated. This mode produces normal sound with no harmonic content.

Bit 6 is the high-pass output (HP). All the frequencies that are above 12 kHz are passed as is, and all the frequencies below 12 kHz are attenuated when the bit is set to a 1. The sound produced with this filtering is tinny.

Bit 7 is the off-on switch for the third voice. When set to a 1, the third voice is disconnected from the audio output circuits.

The lower nybble of the MODE/VOL register is the VOL part. The higher nybble is the MODE section. Bits 0-3 are able to choose between 16 levels of volume that will emanate from the audio output. 0000 will kill volume altogether, while 1111 produces maximum volume. The bit layouts in-between produce varying volume levels

Writing and Reading

The preceding registers were all write-only types. The processor is able to write to all those registers and the individual bits to produce the many and varied sound effects. BASIC, with its POKE statement, can work easily with all the registers. During troubleshooting each register could be POKEd at will to check its operation.

The next four registers are read-only. BASIC with its PEEK function can read all the registers and the bits.

Pot Registers

The next two registers are the ones where the positions of the two POTs, x and y, are stored. The input to the register for POT x is pin 24 of the SID. As the potentiometer is varied a dc voltage is varied. The voltage enters an analog-to-digital circuit which changes the voltage from a single dc value to a relative binary number between 00000000 and 11111111. The zeros represent minimum resistance and the ones maximum resistance of the pot settings. Refer to Fig. 22-9.

The binary values are stored in the POT register x. The value is always the present pot setting and is updated every 512 clocks. POT y works in exactly the same way except for the input that enters

Read only									
54297	7	6	5	4	3	2	1	0	POT x
54298	7	6	5	4	3	2	1	0	POT y
54299	7	8	5	4	3	2	1	0	OSC3 RANDOM
54300	7	6	5	4	3	2	1	0	Envelope 3

Fig. 22-9. All of the registers in the SID are write only except for these four which are read only.

at pin 23. The 8502 can read the two POT registers and always know the positions the pots are set at.

OSC 3/Random Register

This register is able to generate random numbers as one of its abilities. Noise is a true random sound. When the noise waveform is produced in the third voice by setting the noise bit 7 in the control register to a 1, this register stored the binary numbers that represent the random changing noise. The register stores the upper eight bits of the third voice's tone oscillator. These bits are the result of the changing noise waveform being generated. The random numbers can then be read by the processor and used in games. Refer to Fig. 22-9.

Besides producing random numbers, this register can be used for many timing or sequencing purposes. Since this register is constantly changing and recording the upper eight bits of the third voice, different waveforms will produce different types of changing number sets. You've seen how the register bits record the random wave shape. If you set bit 5 to its control register to a 1, the sawtooth waveform is generated by the third voice.

The OSC 3/RANDOM register will then have its bit change in time with the changing sawtooth waveform. Each sawtooth will increment the register by 1. The binary bits will count from 00000000 to 11111111, over and over again. When bit 4 of the control register is set to 1, the triangle waveform will drive the OSC 3/RANDOM register. The reg-

ister will then count in a slightly different way. It will start the same and count from 00000000 to 11111111. Then instead of starting over and continuing its incrementing, it will simply start decrementing the count backwards till it reaches zero again. At that time, it will go back to the incrementing.

The register can record the action of the pulse waveform too. The square-wave type pulse goes from high to low and back and so on. The register counts 11111111 as it is high and 00000000 when it is at a low. Just as the square wave does not have any gradations between the high and low the register does not record any binary numbers that represent in-between states.

The main function of this register though is as a modulation generator. The numbers that can be generated by changing the different waveshapes available can be used in a program to produce sound effects. Combinations of this registers numbers plus the bits generated by other registers can make the audio output sound like sirens, moaning, groaning, and all sorts of interesting effects. Other registers that will work smoothly with the OSC 3/RANDOM register are the pulse width registers, the filter and the oscillator.

ENV 3 Register

The last read-only register is named ENV for Envelope. This register lets the 8502 read the output of the third voice envelope generator. This reg-

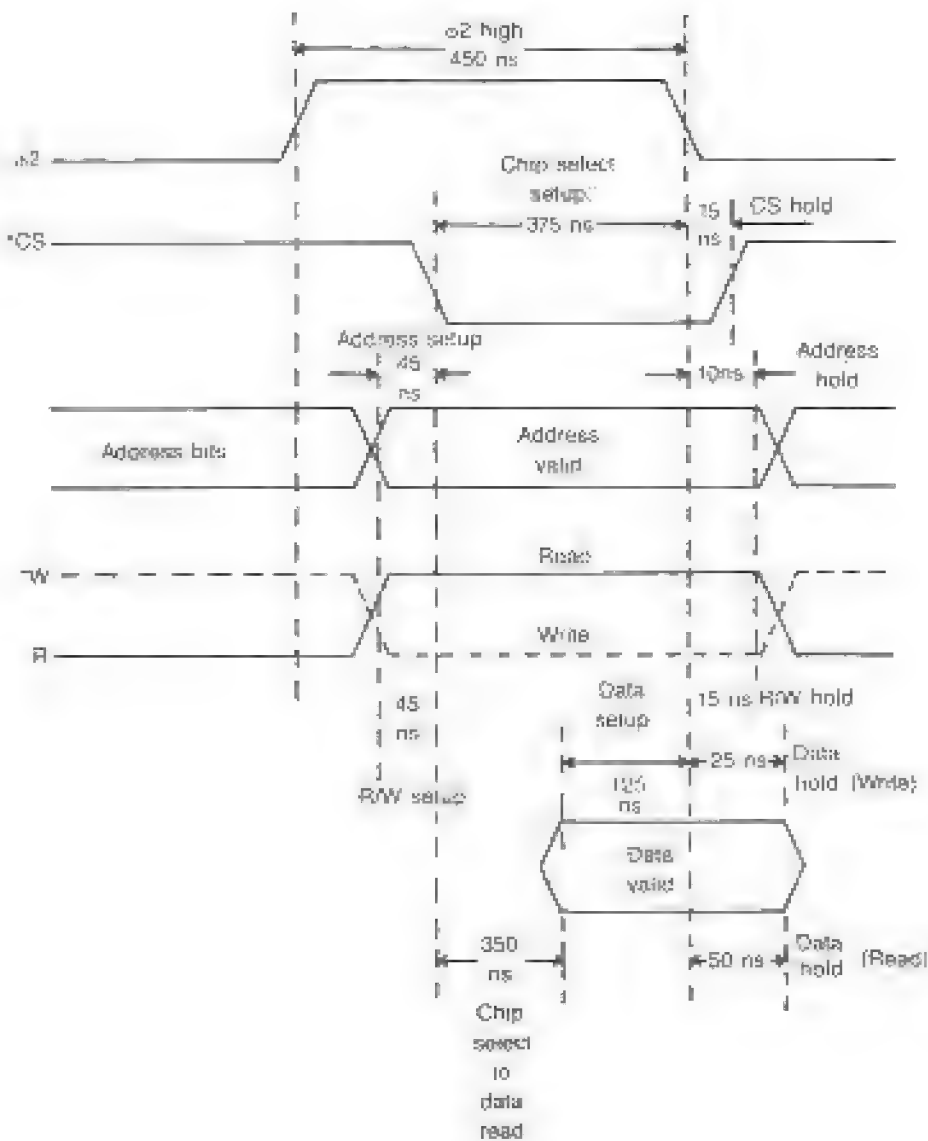


Fig. 22-10. SID is accessed only during the high of $a2$. At that time, $*CS$ goes low, the address bits go through a setup time, a valid time, and a hold time. For a write, W goes low and for a read R goes high. The data is valid near the end of $a2$ and is strobed into the SID's registers as $a2$ falls.

ister only records the output of the envelope generator when the EG is running. The numbers produced in this register can be added to other registers for more sound effects. Refer to Fig. 22-9.

TIMING

The SID is accessed by the 8502 in a very straightforward way. The 29 registers look to the 8502 like ordinary locations. Most of the bits that travel back and forth are used to produce sound effects.

Your Commodore is designed to run at near 1 MHz. This means a full cycle is 1000 ns. The SID is accessed only during the high of $\phi 2$ which consumes 450 ns. When $\ast CS$ pin 8 is made low it must provide at least 375 ns of $\ast CS$ setup time during the $\phi 2$ high. Once the data is accessed the $\ast CS$ stabilizing Hold time needs more 15 more ns. Refer to timing diagram in Fig. 22-10.

As the addresses enter A0-A4, pins 9-13, the bits need a setup time of 45 ns during the high of

$\phi 2$. After the data is accessed, the address bits need 10 more ns as hold time.

The next signal that is also entering SID is the R/ $\ast W$ at pin 7. It runs neck and neck with the address bits and needs the same setup time (45 ns) and the same hold time (15 ns).

The write-only registers receive their contents from the data pins when R/ $\ast W$ is low. The registers need a setup time of 125 ns to settle the bits into place. Once the bits are stable, a hold time of 25 ns is needed to be sure.

The read-only registers place their contents onto the data pins when R/ $\ast W$ is high. The total access time from when the chip is selected till the data is placed on the system data bus is 350 ns. Then it takes a data hold time of 50 ns to stabilize the bits in the data system.

TESTING

You can check out SID quickly by POKing the numbers in Fig. 22-11 into the SID. It will produce

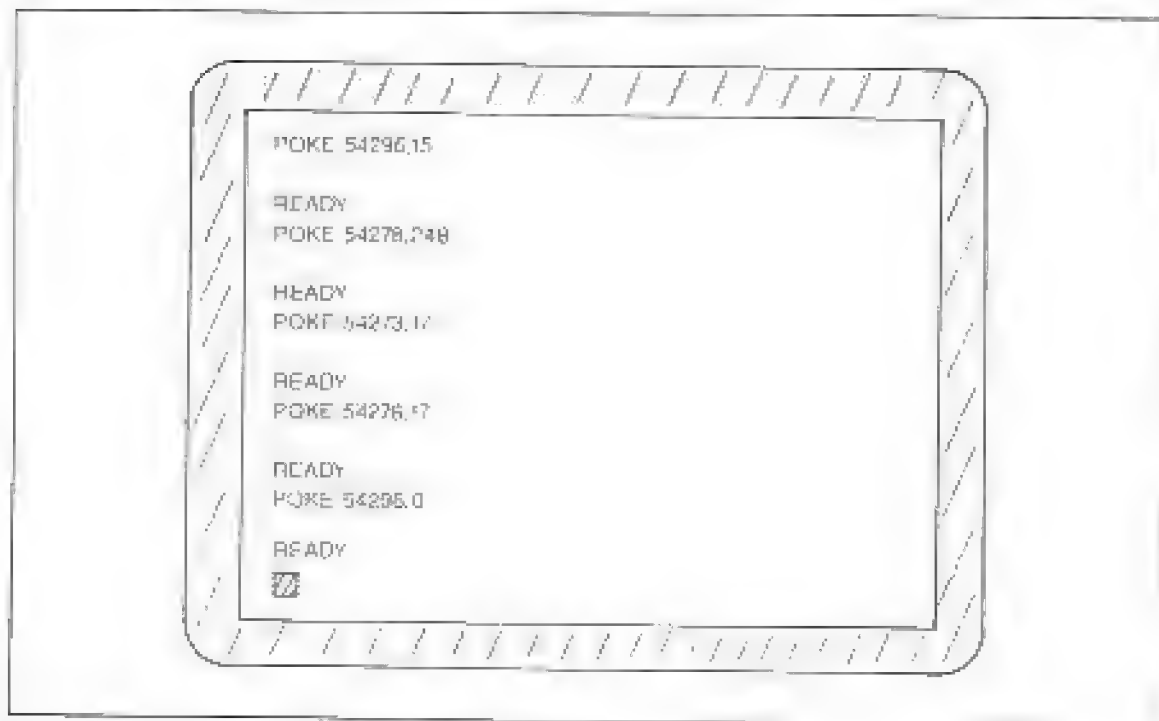


Fig. 22-11 You can check out SID quickly with these POKEs.

a continuous tone from voice 1. The first POKE sets the volume control register. The second POKE arranges the A/D S/R properly. The third POKE sets the note in the high frequency range. The fourth POKE sets the waveform to a sawtooth.

In a good SID, the note should come on loud

and clear. If it does not the SID has trouble. To turn off the note, POKE 54296, 0. This shuts down the volume register. To check out the other voices of the high frequency register, simply substitute the other register addresses.

23. Inputs and Outputs

When you sit at your computer, with your fingers on the keyboard and your eyes watching the TV display, you and the C128 become a closed circuit. Your fingers input data from your brain. The keyboard codes your data into binary bits the digital circuits need to work with. Your fingers on the keyboard are your input into the innards of the digital circuits. You are an external device inputting to the computer.

Your binary coded keystrokes are fed to CIA1, are passed to the processor, are then processed on through to VIC or the 8563. If you are working in 40 column, VIC sends the output. In 80-column the 8563 does the honors. Whichever mode is used, a picture of the key character appears on the TV display. You read the screen, and you can tell from the way the data appears how you should continue your computing. The data on the screen thus becomes an input, through your eyes, back into your brain. The C128 and you are a system. Either the computer is a peripheral to you or you are a peripheral to the computer, according to who is in control.

Besides interfacing with you, the C128 is able to receive inputs from joysticks, paddles and a light pen. It is able to output data directly to the cassette, disk drive, printer and modem. The C128 is also able to connect external ROM cartridges by means of the expansion port and other devices into the memory map.

As Table 23-1 shows, there are eleven port plugs, an ac internal connector, and the LED plug, to handle the inputs and outputs. They are numbered CN1 through CN13. Connector CN1 is the 44-pin expansion port. It connects cartridge ROMs to all the internal bus lines. Connector CN2 is an I/O slot that controls the cassette motor as well as the tape reading and writing. Connectors CN3 and CN4 are two 9-pin input control ports that are needed to input joystick positions and pushbutton impulses. Connector CN5 is the keyboard I/O plug with 25 pins. It inputs to the CIA1 directly. CN6 is the valuable Serial Port that does input and output work. It connects data to the printer and/or to a disk drive, either single- or double-sided.

Table 23-1. Of the thirteen connectors with the CN designation, Only ten are available to the user. Connector CN7 and CN12 are internal and CN13 is the LED pilot bulb plug.

Connector Name	Description
CN 1	44-Pin Expansion Port
CN 2	12-Pin Cassette Port
CN 3	9-Pin Controller Port 1
CN 4	9-Pin Controller Port 2
CN 5	25-Pin Keyboard Port
CN 6	5-Pin Serial Port
CN 7	5-Pin Internal Serial Bus
CN 8	6-Pin Composite Video Output
CN 9	24-Pin User Port
CN 10	9-Pin RGBI Video Output
CN 11	5-Pin ac Adapter Plug
CN 12	5-Pin Internal ac-dc Connector
CN 13	3-Pin LED Light Plug
RF Output	72 Ohm Video Plug

Connector CN7 does not connect to the outside. It is the internal serial port. It is a way station to help the actual serial port, CN6, with its I/O duties. Connector CN8 is the composite video output directly from VIC. It is input to a TV monitor to display 40 columns.

Connector CN9 is the 24-pin User port. It allows connections to many peripherals. It connects into CIA2. A printer, modem or another computer like a C64 or a C128 could be connected to CN9 with the proper interfacing. Connector CN10 is the 80-column output plug from the 8563. It will output either an RGBI or a monochrome TV signal—both in 80 columns.

Connector CN11 is the input plug from the ac adapter power supply. It is covered in detail in the next chapter. The final output is the RF Modulator circuit. It is a little metal box containing the transistorized RF oscillator and amplifiers. It has one output plug resembling a phono jack where the 72 ohm cable from the C128 to the home TV antenna terminals are connected. More detail is given at the end of this chapter.

CHECKING OUT I/O TROUBLES

When an I/O trouble happens, the first step in the troubleshooting is deciding whether the prob-

lem is being caused by the C128 or the peripheral that is not working. This can be easy.

If you have a spare peripheral or a second compatible computer that works okay, then you can make a substitution. For instance, if the cassette stops operating properly, then try a known-good spare cassette or try the suspect cassette on a second computer. If the spare cassette works on your C128, then the trouble is in the original cassette. Should the second cassette not work either, chances are good that the C128 internal cassette circuits are malfunctioning.

Besides the obvious substitution method, a diagnostic program like the 64 Doctor, mentioned in an early chapter, is handy. You could run the 64 Doctor in the C64 mode on the C128. It will run accurate tests on the keyboard, the TV monitor, the three voices of SID, the single-sided disk system, the printer, cassette and the joysticks. It tells you whether the device and the ports are operating correctly or not. The 64 Doctor, or a program like it written for the C128, can be useful in isolating a defective peripheral system and telling you if the peripheral is bad or if the C128 is causing the problem. Also, don't be afraid to use these principles and write yourself some little diagnostic programs to check out the I/O systems. These programs work well because only a peripheral is acting up and the rest of the computer can run the program.

CN1 Expansion Port

The expansion plug is the large 44-pin connector on the upper right-hand corner of the board. It has 22 connections on the top side and 22 more on the bottom. This port is the one with the most versatility. It connects the user to almost every important signal in the machine. The plug has connections to address and data bus lines. Also available are the 1 MHz, R/W, +IRQ, +NMI, +RESET and others, shown in Fig. 23-1. As a result of this access, this port is a good place to test for the presence of most of the signals that are supposed to be coursing through the computer.

To aid you during point by point checking, Fig. 23-2 shows the female edge connector has the up-

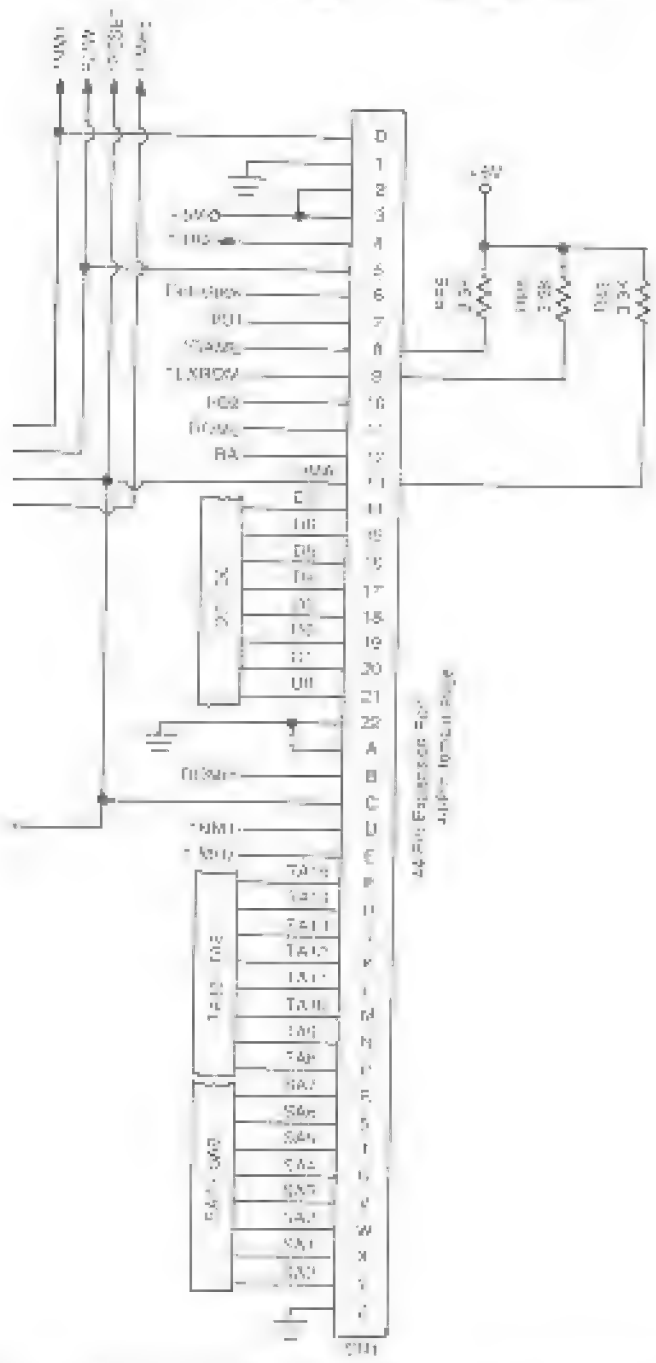


Fig. 23-1 The 44-pin expansion port has connections to practically every important bus in the computer. A cartridge installed in the bus can take control of the computing.

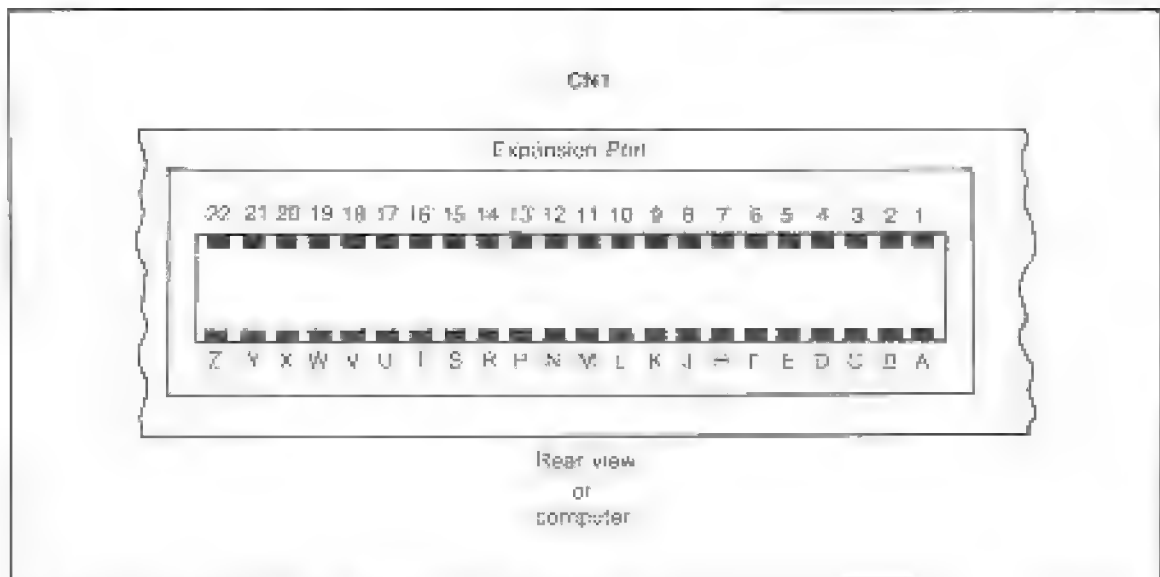


Fig. 23-2 The 44 pins of the port are numbered 1-22 on the top and labeled A-Z on the bottom.

per 22 pins numbered 1-22. The lower edges are labeled A through Z with G, I, O and Q missing. In Fig. 23-1 are four ground connections: 1, 22, A and Z. Pins 2 and 3 are the +5 volt supply. They are filtered with a 10 mF, 25 working volt capacitor.

The eight data bus lines are connected to pins 14-21. The address lines are connected at F through Y. Pins R-P are attached to the translated address bus, TA15-TA8, while R-Y are connected to the shared address bus lines, SA7-SA0. Line R/*W is at pin 5 and *IRQ is on pin 4. The bus lines should show all logic probe pulses, and the two control lines are normally held high. To turn on *IRQ it must be forced low. Pins 8 and 9 are *GAME and *EXROM. They are external device inputs to the MMU and PLA. They are held high when not in use. Pins 11 and 8 are ROML and ROMH outputs from the PLA and are also held high when not in use. They go low when active. Pin C is *RESET, an input. It is also held high when unused and goes on when forced low. *NMI is an unused input or output connection at pin D. Pins 7 and 10 are both outputs from the decoder chip U3, a 74LS138.

One MHz is on pin E. Pin 6 is the video dot clock. A lot of the system timing is a result of that frequency. Pin 12 is the bus available signal, BA,

that originates in VIC. It is held high till VIC takes control of the bus lines. It goes low three cycles before VIC goes into action and stays low till VIC is finished accessing and displaying its TV signals.

Pin 13 is the direct memory access, DMA. It is held high through a 3.3K pullup resistor. An external processor can be connected to this interface and force this line low during a system clock low. When that happens, the R/*W line, the address bus and the data bus become three-stated. The outside processor can then time up with VIC and run the C128. When not in use, it should be high.

CN2 Cassette I/O Slot

The cassette slot, in Fig. 23-3, looks like it has 12 pins but electrically they are only six. The corresponding pins at top and bottom are tied together. The top pins are 1-6 and the bottom pins are A-F. This provides six pins called 1-A, 2-B, 3-C, 4-D, 5-E and 6-F, as seen in Fig. 23-4. When you test A you are testing 1 simultaneously. A-1 is the ground connection and B-2 is the +5 volt supply.

C-3 is the motor switch, E-5 is the write output line and F-6 is the cassette sense. These lines were all discussed in the 8502 chapter. They are 8502 pins 25, 26 and 27 which are P5, P4 and P3

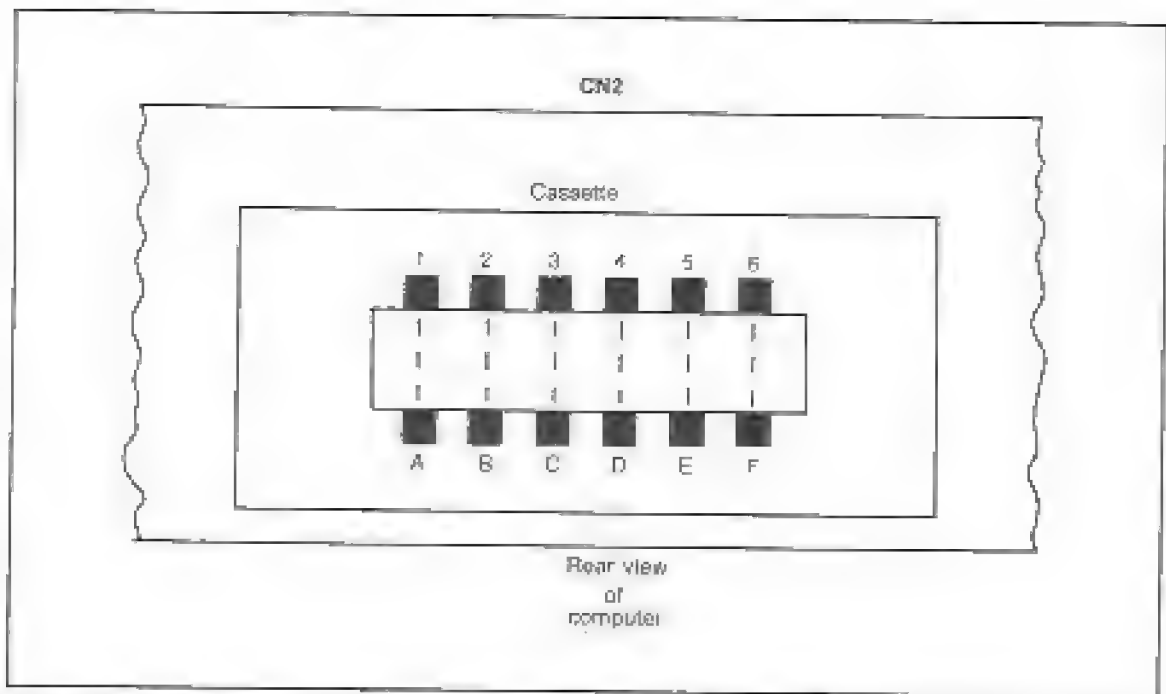


Fig. 23-3. Connector CN2 is a 12-pin slot but, because the top and bottom layers are connected one-for-one, only six individual connections are evident.

respectively. They are connected to the special 8502 internal I/O register. The setting of the register bits 3, 4 and 5 decide what action these lines are to take. The 8502 test point chart shows their normal standby states. Line P5 is high, P4 is high and P3 is held low.

The only remaining pin is the cassette read D-4. It is held high by a 3.3K pullup resistor. When the processor wants to read from the cassette, it forces D-4 low and the read will take place. D-4 connects to pin 24 of the keyboard CIA1. Pin 24 is $\overline{\text{FLAG}}$. When it is pulled low, it signals the chip that a data transaction is about to take place.

CN3 and CN4 Control Ports

When a joystick, paddle or light pen refuses to work, either the device is broken or the computer has a defect. If available, a known-good spare should be tried. If the replacement restores the activity then the old one was defective and the trouble has been found. Should the spare one still not operate, then the computer becomes suspect. When the computer

is the trouble, begin troubleshooting at the inoperative control port. Nine pins are on each control port. Each port can connect to a joystick or paddle. Control port 1 can also have a light pen plugged into it. The ports are shown in Fig. 23-5.

On each port, in Fig. 23-6, pins 7 receive +5 volts and pins 8 are both ground. Pins 1-4 are the joystick inputs. Port 1 connects its joystick inputs to the port pins PB1-PB4 of CIA1. These are the same pins that rows 1-4 of the keyboard are connected to but since the joysticks and keyboard are not supposed to operate together, there is no apparent conflict.

Control Port 2 connects to CIA1 pins PA1-PA4 in the same way. These are also keyboard connections from columns 1 to 4 and no conflict is indicated. The buttons on the joysticks are from pin 6 and go to PB0 and PB1 of the CIA1. The four joysticks and the buttons are all switches. The position switches are used to direct the image on the screen and the buttons are fire switches. The switches are connected to the lower five bits of the port registers.

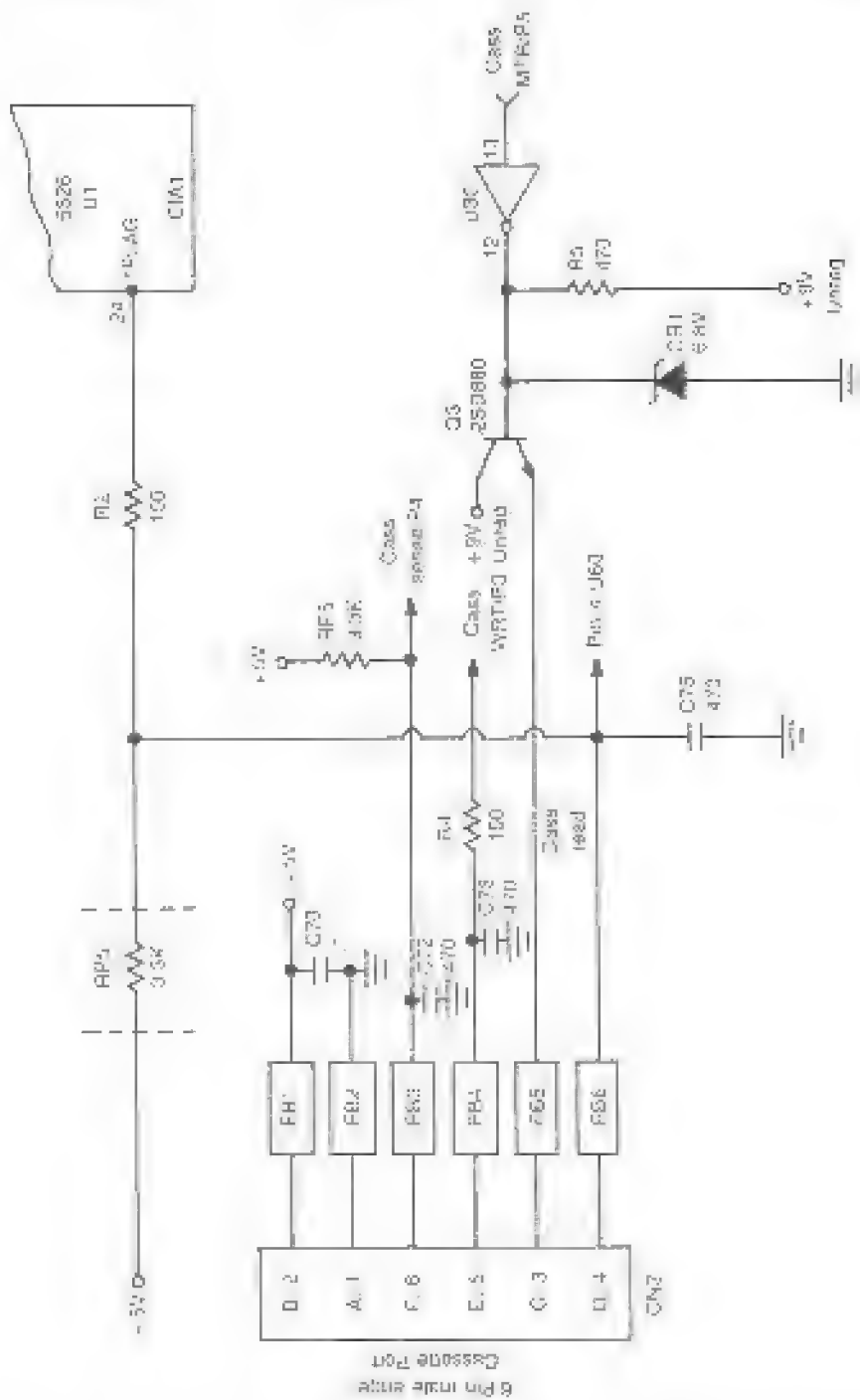


Fig. 23-4. Connector CN2 wires a capacitor to power, a read line, a write line, control and motor circuits.

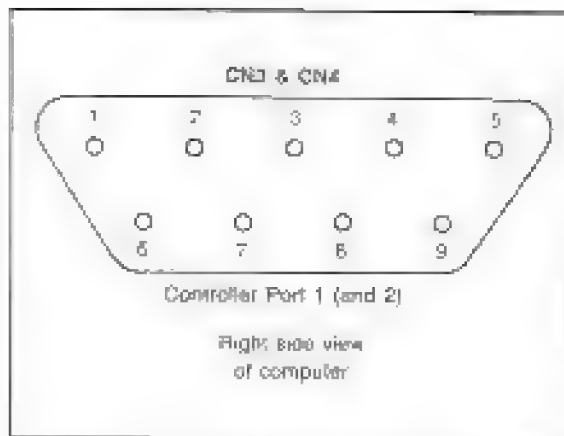


Fig. 23-5. The two control ports, CN3 and CN4, are plugs on the right side of the computer with these pins.

in CIA1. The switches are held high till a direction is chosen or the button is pressed. Then the switch shorts the bit to a low.

The quick way to check out these ten inputs on the two ports is with the logic probe. A high should be present when the computer is first turned on and the keyboard is untouched.

The ports each have pin 9. Control Port 1 outputs to SID signal pot *x* and Port 2 outputs pot *y*. These two inputs are checked easily with the logic probe. They are both lows and remain low all the way through to SID.

The light pen shares pin 6 in Port 1 with the fire button connection. If the button is working, then a good light pen will too. Test for the standby high the button uses to operate with.

CN5 Keyboard Port

The C128 keyboard has its characters laid out in what is known as the QWERTY arrangement. This means the first six letters, from top left to right spell QWERTY. This is the conventional typewriter layout. When you learn to touch type you commit the letters to reflex. In addition, many more keys, and even an individual numeric keypad, are on the keyboard. This arrangement is for your convenience. The computer doesn't see the keyboard characters in that way.

The C128 sees a grid of eight rows and 11 columns. In addition, the SHIFT LOCK, RESTORE,

40/80 and CAPS LOCK keys are not included in the grid but are inputs to individual circuits. There are eight switches and 80 intersections in the grid. This gives a total key count of 93.

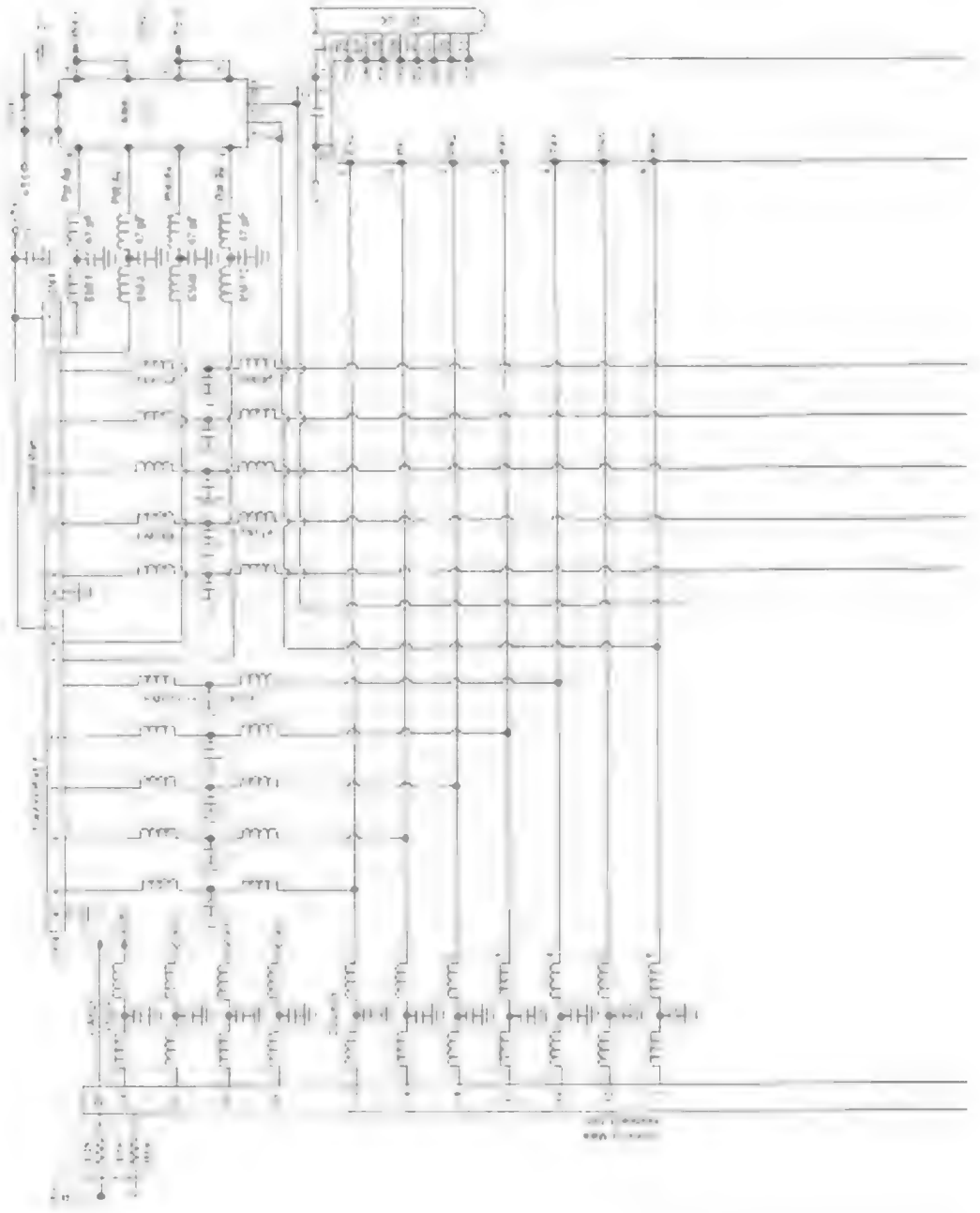
If you look at the schematic of the keyboard circuit, back in Fig. 5-7, then you'll see the layout. A separate switch is in every row of the grid. The rest of the switches are at the bottom of the page. If you press one of the switches, then an impulse will travel out of that particular row and enter the keyboard port plug CN5 at one of the row pins. For instance, pressing RETURN will send an impulse out to pin 11 of CN5. Pressing 40/80 will send an impulse out to pin 24.

At the 80 intersections though, the pressing of a key does not produce as straightforward a result. Every time one of the keys is pressed, one of the rows and one of the columns are shorted together. Note the bottom right circle inset of a switch that is below every intersection key. When the key is pressed, the switch shorts the row to the column.

Connector CN5 is the port on the printboard at the bottom right-hand corner. It is not accessible unless the case and shield are removed from the printboard. The keyboard is plugged in here. Connector CN5 is a 25-pin male DQ port. See Fig. 23-6. Eight column connections and eight row connections are shown. The column connections are attached to the PA0-PA7 port pins of CIA1. The row connections are attached to the PB0-PB7 port pins of CIA1. The row connections are inputs to CIA1.

The column connections are outputs from CIA1 to the keyboard. These output pins send pulses to the keyboard. The pulses, moving quickly in comparison to even the fastest typist, are strobe pulses. They constantly scan the columns, pin by pin, with a strobe pulse generated by the system clock. The strobe starts with column one and checks it for short circuits. If there is no short circuit at column one, then the pulse checks column two. The pulse keeps testing each column in turn till it arrives at a column that has one of its rows shorted to it due to you striking a key on the board.

The eight rows are connected to eight pins on the CIA1. When a strobe pulse locates a row short, it causes that row to input a pulse into the chip. The



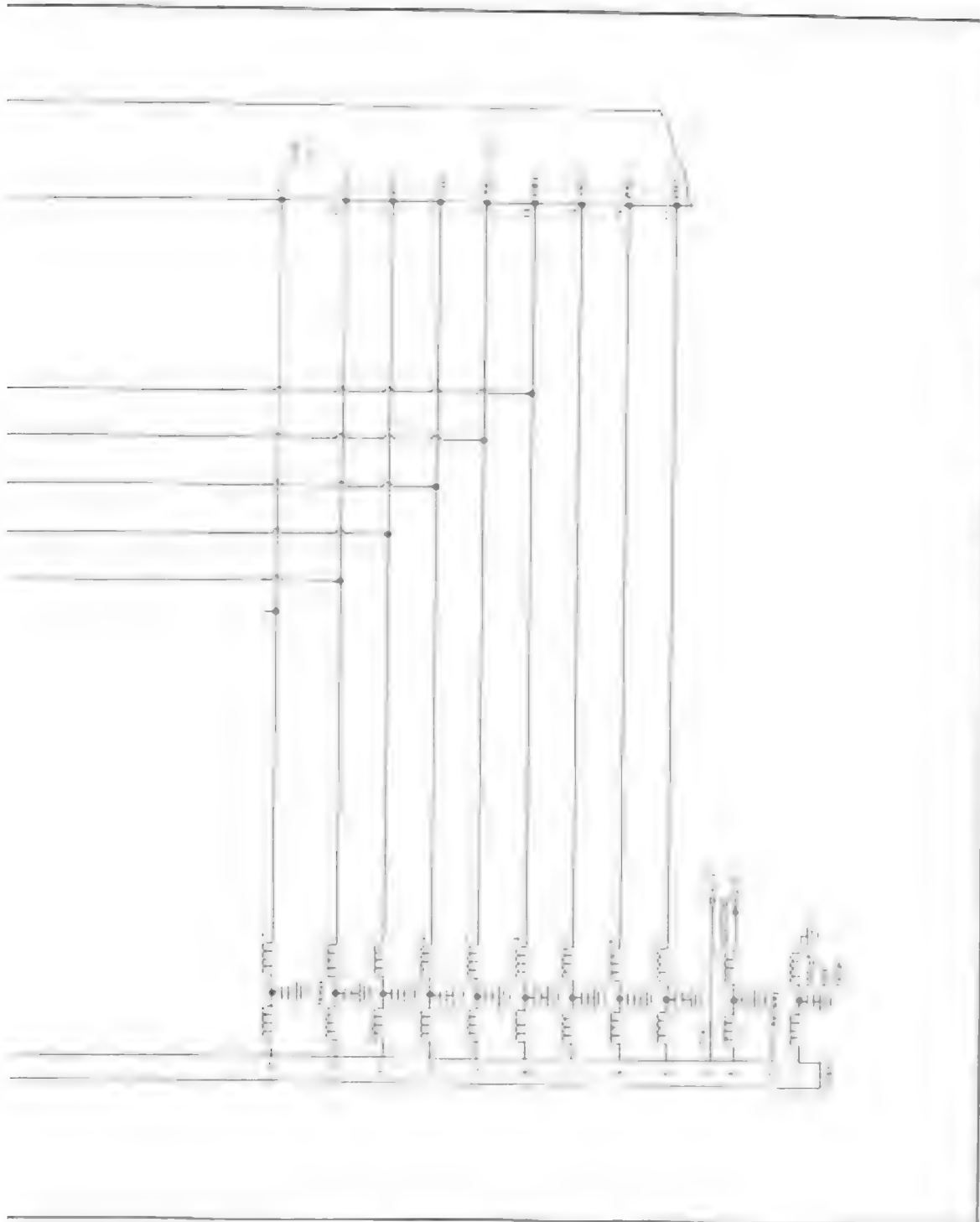


Fig 23.4. Internally, the control pins share input lines into CIA1 with the keyboard. No conflicts arise because peripherals like joysticks are not used simultaneously with the keyboard.

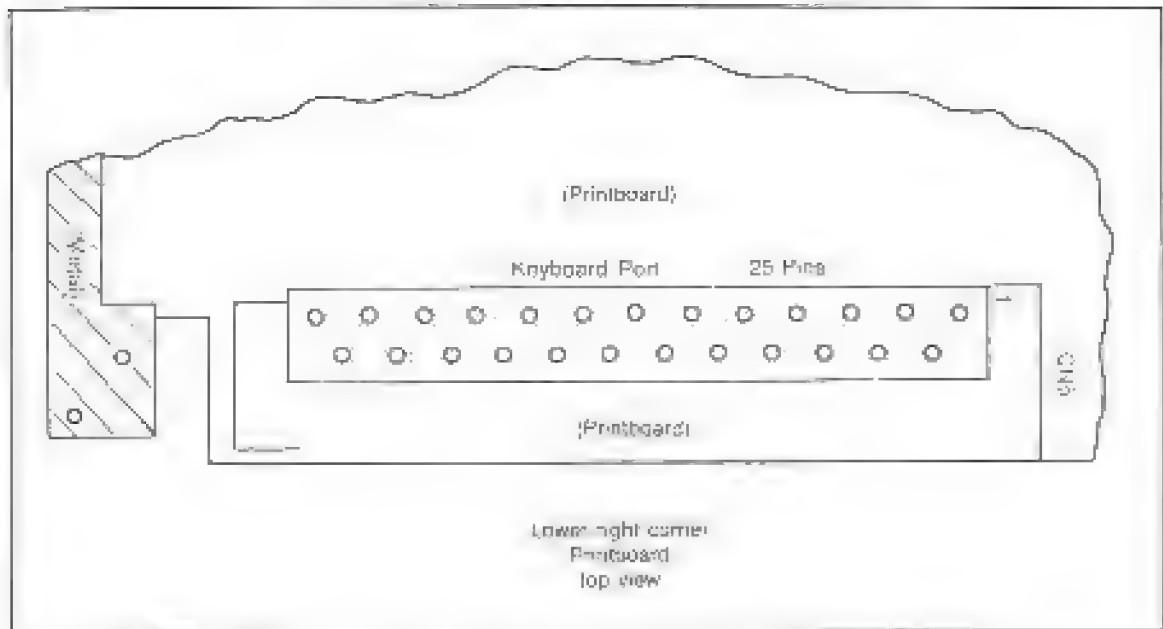


Fig. 23-7 The keyboard port has 25 pins mounted on the printboard inside the case.

correct row is thus registered at the input. At the same time, the strobe pulse has been keeping track of which column the row was shorted to by you pressing the key. With the knowledge of which one of the intersections is shorted, row to column, the computer knows which key on the keyboard has been pressed. CIA1 then outputs the code of the pressed key over the data bus to the processor. The processor takes it from there.

CN5 is the port between the keyboard and CIA1, seen in Fig. 23-7. It handles all the keyboard outputs and in addition supplies the keyboard with the strobe pulse inputs to scan the columns.

CN6 and CN7 Serial Ports

The Serial Port, Fig. 23-8, performs both inputs and outputs; it has six pins. Pin 2 is grounded and pin 6 is connected to the external reset, +EXTRES, on the user port, CN9. The remaining four pins perform the serial transmission of data. The data can travel from the C128 out to a device like a printer or both ways when it is used for peripherals such as the disk drive. Figure 23-9 shows the arrangement.

Pin 1 is the serial-service request, +SRQIN. The +SRQIN line can be hooked up to a number of external devices. +SRQIN is held high. If a device wants service, then it forces +SRQIN low and the line responds.

Once pin 1 goes low, the C128 will react. It brings pin 3—the serial attention in/out line, ATN—low. If there are any other devices on the line, then they are alerted. The devices on the line can do three things. They can listen, talk or control. The devices—items like a disk drive, graphic printer or others—are all connected on the same line. They can all listen at the same time, but only one can talk at a time. The computer controls the one it wants to talk.

When pin 3 is then brought low, all devices listen and the one instructed to talk will respond. The data that the device sends to the C128 will arrive in serial fashion: a bit at a time; over pin 5; serial data in/out; DATA. Pin 4 carries a clock signal, CLK, to sync the timing of the external devices with the processor. Pin 4 is called serial clock in/out, CLK.

CN7, Fig. 23-10, is the Internal Serial Bus. It is a convenient post connection network that helps

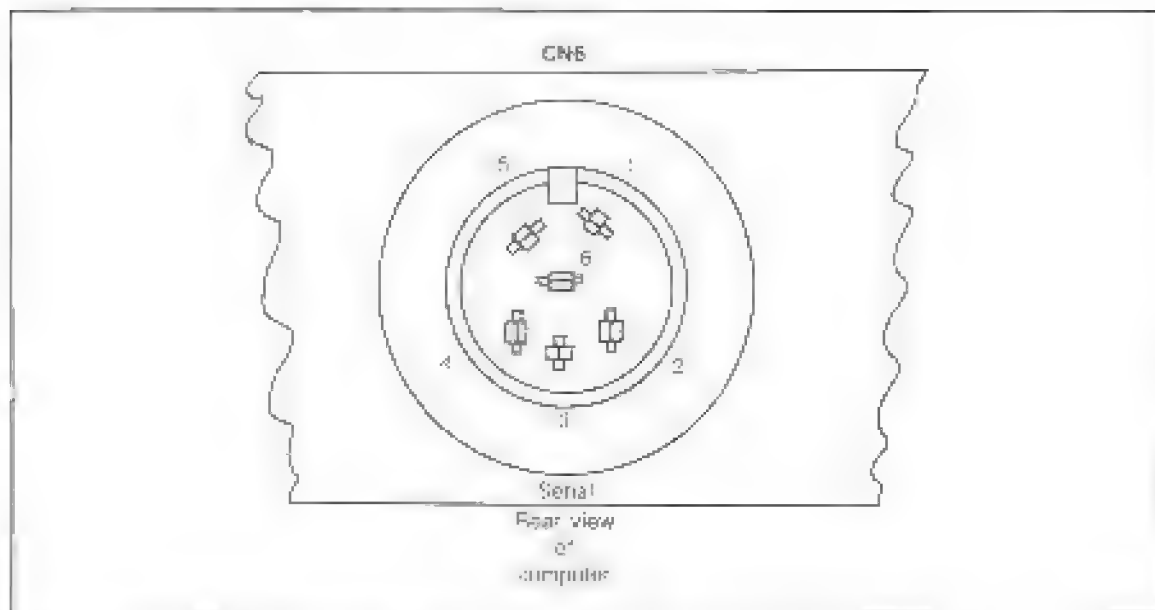


Fig. 23-9 Connector CN6, the serial port, has six pins available on the back of the computer.

the CN6 port handle the data that passes between the C128 and its serial peripherals; CN7 is a form of way station where the signals can get routed and even receive some processing by nearby gates.

In Fig. 23-11, it can be seen that pins 1 and 5 of CN7 connect to some buffers, triggers, NANDs and an AND. This network of gates logically process signals between CN7 and the user port, CN9, discussed later in this chapter.

Pins 2, 3 and 4 are connected directly to the serial port CN6. Note that CN7 has one more pin than CN6; pin 7. It receives a signal called *DRESET. This is a reset connection and is not involved with CN6.

RF MODULATOR

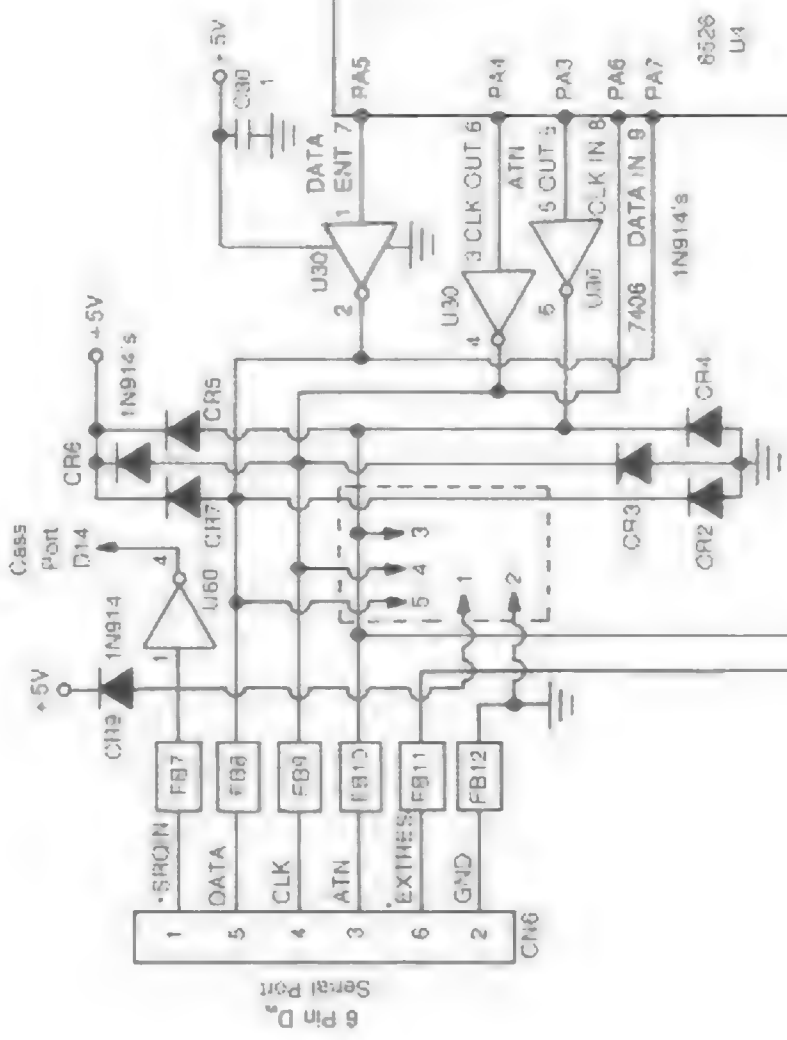
In Chapter 20 and the Master Schematic, the VIC system is shown. Coming out of VIC pins 17 and 16 are the SYNC/LUM and CHROMA signals. They connect directly into pins 2 and 4 of the RF Modulator box. In Fig. 9-9 are the circuits that are found in the box. The circuits are based around four transistors. The SYNC/LUM signal is input to a 2SC458 npn amplifier. The CHROMA signal to a second 2SC458. The two signals are amplified and

then joined together through some coupling resistors, capacitors and a small coil. The combination output is then placed into the RF OUTPUT line where the signals are sent through a filter network and then placed at the RF OUTPUT plug.

Meanwhile, at the top of the schematic diagram, the audio output from SID enters the circuit through pin 8 of the box. The audio is sent through a 10 mF capacitor and some coupling resistors and capacitors, into a 2SC460 audio amplifier. The transistor output is then coupled into the same RF Output line and joins the video signals at the RF Output plug. The video and audio are ready to emerge from the plug except for one more item.

At the bottom of the drawing is a circuit based around another 2SC460 transistor. The transistor is operating as an RF oscillator and can run at either the Channel 3 or 4 commercial frequency. The channel select switch chooses the desired frequency. The switch is located at the back of the computer.

The RF signal developed in the oscillator circuit is also coupled directly into the RF Output line. It mixes with the video and audio and the result is a signal that closely resembles the TV signal a TV station transmits. For instance, if the switch is set



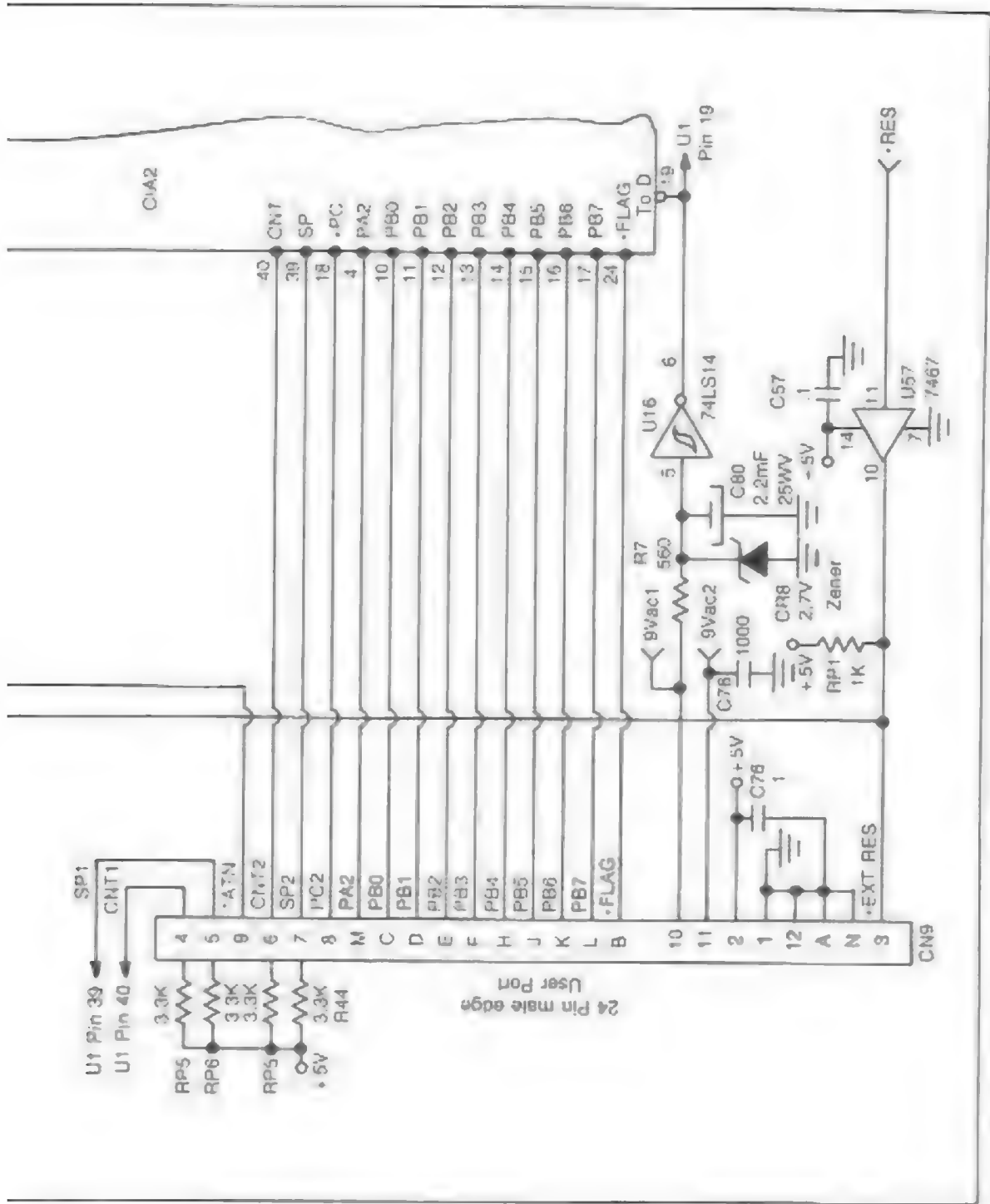


Fig 23-8 The serial port is wired to pins on CIA2

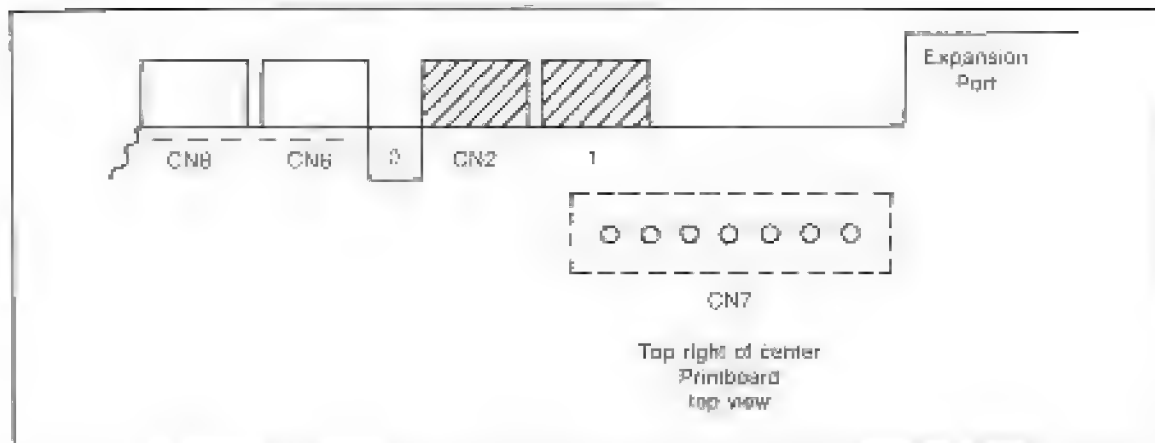


Fig. 23-10. Connector CN7 is the Internal Serial Bus. Physically it is simply a row of solder connections on the printboard.

at Channel 3, then a color TV signal is output that varies between 60 and 66 megahertz. This carrier wave modulated with video, color and audio can be obtained at the RF Output plug and connected to the antenna terminals of any commercial TV. A strong signal will be seen on channel 3 of the TV.

Besides outputting the TV channel 3, the RF Modulator box outputs all the signals, except the RF oscillator and the audio independently. A tap is connected to the Sync/Lum-Chroma output between a 270 ohm resistor and a 150 pF capacitor. This tap picks up some of the signal and outputs it from pin 5 of the box. This signal contains all the video, color and sync. It is a composite color TV signal without an RF carrier.

Across the 150 pF capacitor is the emitter of the 2SC458 and connects to a voltage divider consisting of an 82 and 150 ohm resistor to ground. At the center tap of the two resistors, pin 7 of the box is connected. At this juncture the color signal can be obtained.

Back in the emitter circuit of the other 2SC458 is another tap through a 100 ohm resistor. This connection picks off the sync/luminance signal and connects it to pin 6 of the box. No output comes from the box for audio. Only composite TV from pin 5, color from pin 7 and sync/luminance from pin 6 are available, besides the RF Output for Channel 3 or 4. Figure 23-12 illustrates the RF Modulator circuit connections.

CN8 Audio-Video Port

The RF Modulator connects to CN8, called the Composite Video Connector, in Fig. 23-13. The name implies that composite video is its output but there are other outputs too. In Fig. 23-12 the composite video, coming from pin 5 of the RF Modulator, only takes up one CN8 pin: pin 4. At this connection, you can obtain a color TV picture with sync that can be connected into a composite TV monitor, without need for any other signals.

A number of other signals, however, come out of CN8 and even one audio input connection. Coming out of pins 1 and 6 are SYNC/LUM and CHROMA (also called COLOR). These two signals together comprise a composite color TV signal. They are needed separately to drive a TV monitor known as a Direct Monitor. If you do not use a direct monitor, these two signals will probably not be used at all.

Pin 3 of CN8 receives the audio output from SID directly. This signal is used for monitors or for any audio amplifier. The signal is pure audio with no other signals mixed. While audio is placed into the RF Modulator box to be placed on the Channel 3 or 4 carrier, that audio is not tapped off and used in the CN8. Connector CN8 gets audio straight from SID into pin 3.

Pin 5 is reserved for an external audio input. The audio can enter CN8 pin 5 as an input. It then passes through a ferrite bead, FH15, over top of a

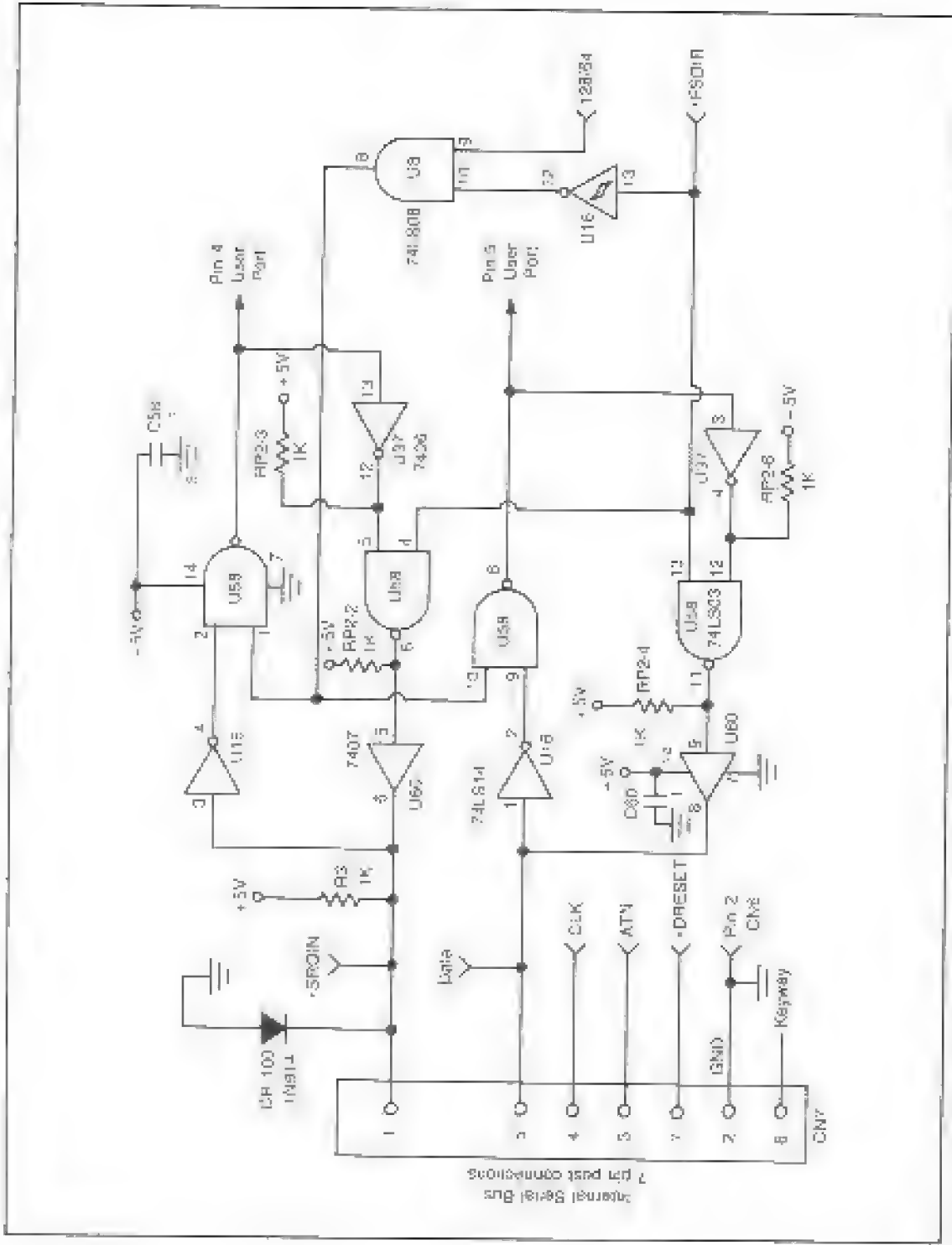


Fig. 23-11. Connector CN7 is a 7-way ribbon for CN6; the external serial port. It uses CN6 to buffers, triggers, NANDs and an AND.

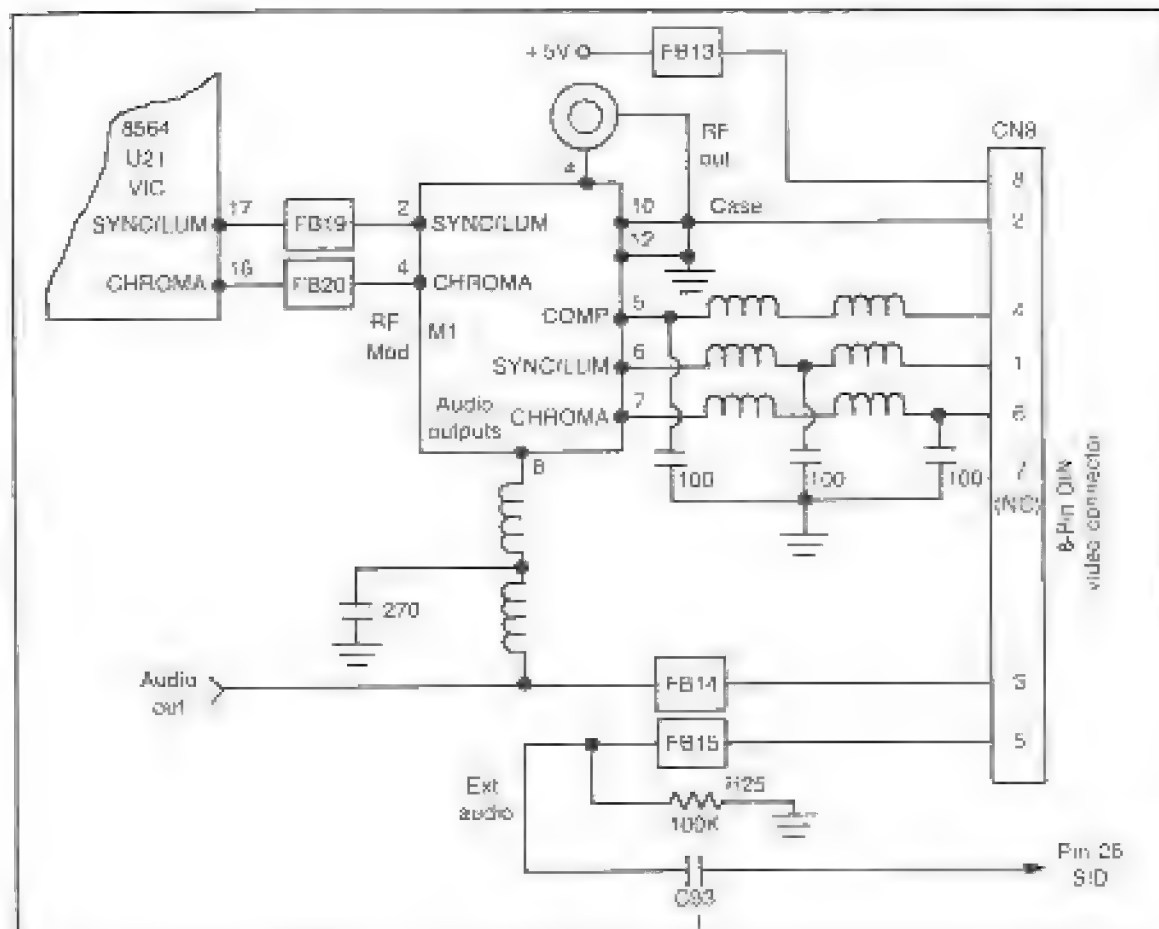


Fig. 23-12. The RF Modulator output plug comes right out of the back of the metal RF box. Coming out of the side of the box are the video signals to CN8.

100K resistor to ground and through a .1 capacitor to pin 26 of SID.

Connector CN8 winds up with pin 2 grounded to the RF MOD case. No connections to pin 7 are made. Pin 8 connects to +5 volts through FB13.

CN8 User Port

The 24-pin user port is split into two 12-pin sections, one on top and the other across the bottom. The top pins are numbered 1-12 and the bottom A-N as seen in Fig. 23-14. The port has a number of control lines connected to it but one of the main functions is to attach a peripheral to the PR output port of CIA2, shown in Fig. 23-9.

Eight lines connect from pins C-L to the PR0-PR7 pins of the CIA2. These lines are for both input and output jobs. Two lines control a handshaking operation. They are pins B, +FLAG, and pin M. PA2. Pins A and N are grounds.

On the top side, pins 1 and 12 are grounds while 2 connects to +5 volts. Pins 10 and 11 are the system input points for the 9 volt ac with a positive phase and 9 volt ac with a negative phase. They come from the power supply transformer.

Pin 3 is attached to +EXTRES, the reset circuit. This pin is usually held high. If you force it low by shorting it to ground, then the processor will restart and reinitialize the entire machine. This is

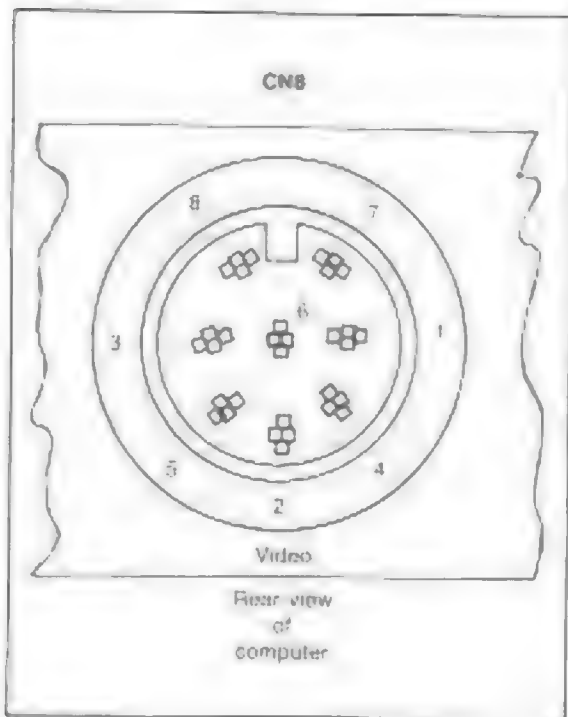


Fig. 23-13. Connector CN8 is an 8-pin plug that outputs video signals. Out of pin 4 comes a composite color TV signal that will drive a TV monitor. Pins 1 and 6 output the Chroma and Sync/Luminance signals to drive a special direct monitor.

a good service test. The C128 will be restarted but if there is any data in memory it should still remain if the computer is operating okay.

Pin 8 is another handshaking control line that goes to $\cdot PC$ of CIA2. Pin 7 is the serial port line that can transfer data to and from pin 39 of CIA2. It is a valuable and much used I/O connection. Pin 6 is the Serial Port Counter line for CIA2 that works with Pin 7.

Pin 5 is the counterpart serial port line to the keyboard CIA1. Pin 4 is the Serial Port Counter line that also goes to CIA1. The user port therefore has two useful serial ports in its repertoire—one on each CIA.

Pin 9 does not work with the other pins. It connects to the $\cdot ATN$ line of the serial port, CN6. It works with PA3 in the PA0-PA7 port register, the other one in CIA2.

The user port is needed mostly to handle the I/O for modems. The port can do many other jobs too. For instance it can perform the I/O duties in communicating with another computer.

CN10 RGBI Port

The RGBI connector, Fig. 23-15, is a 9-pin D type. RGBI stands for Red, Green, Blue and Inten-

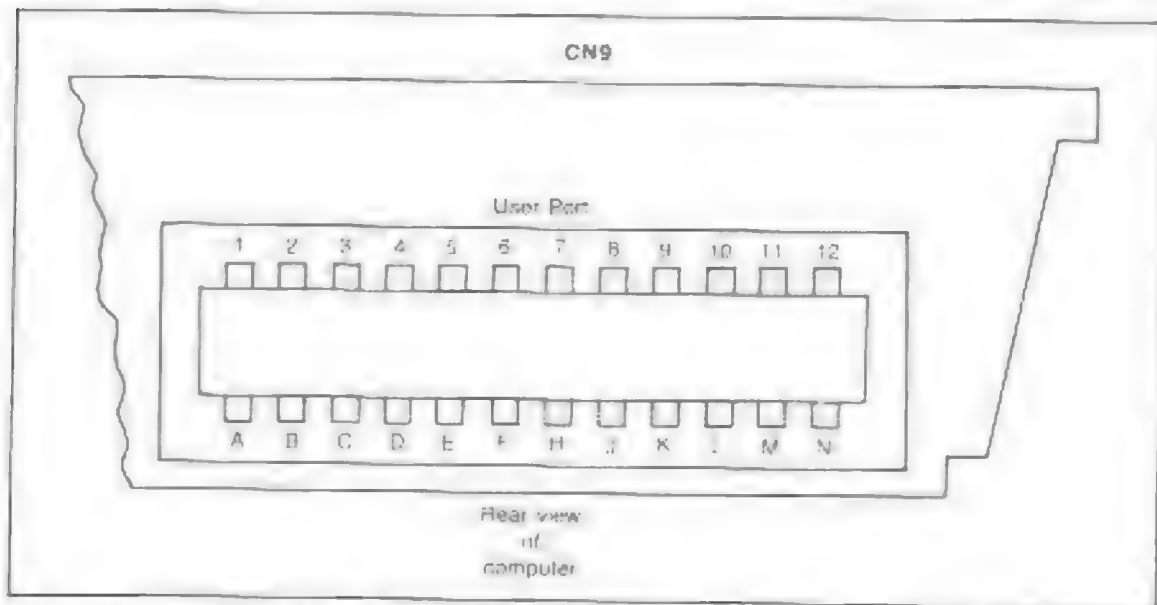


Fig. 23-14. Connector CN9 is the user port. It is numbered 1-12 on the top and labeled A-N on the bottom.

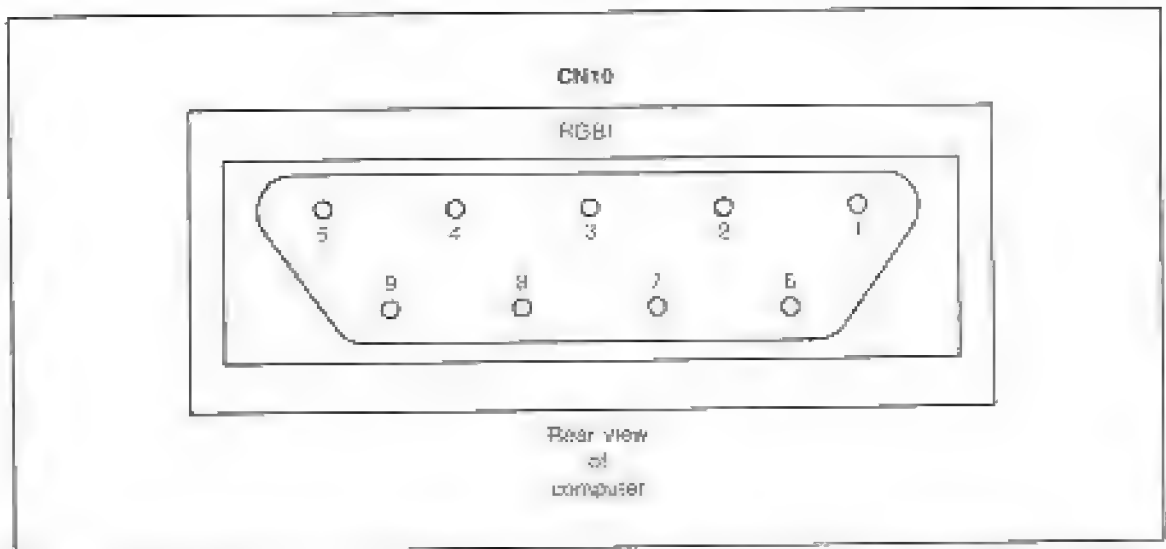


Fig. 23-15. Connector CN10 is the 9-pin RGBI port. It outputs RGBI and a composite monochrome 60-column display.

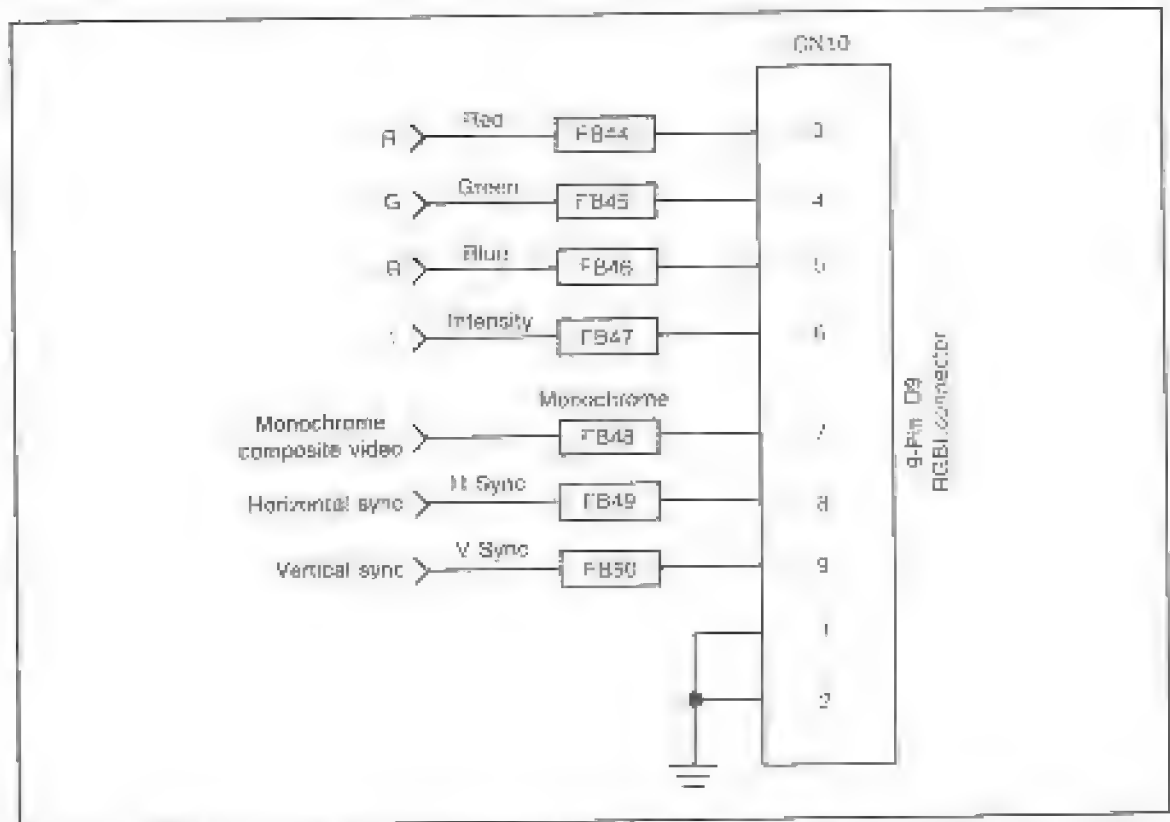


Fig. 23-16. Pins 3-6 output RGBI and pins 8 and 9 the accompanying horizontal and vertical sync signals. Out of pin 7 comes the composite monochrome video.

sity. This is an 80-column output and handles the signals the 8563 generates.

Besides outputting R, G, B and I from pins 3, 4, 5 and 6, as Fig. 23-16 shows, CN10 also sends out the horizontal sync and vertical sync from pins 8 and 9. In order to display the TV picture, CN9 must supply six circuits in the RGBI monitor. These circuits in turn do the following.

The three color signals, R, G and B drive the three electron guns in the color CRT. The Intensity signal determines the intensity of the electron in the beam that leaves the guns. This determines the brightness or luminance on the screen.

In the monitor are horizontal and vertical sweep circuits that produce the scan lines on the screen. The horizontal sync signal locks the horizontal scan frequency. The vertical sync signal locks the vertical sync frequency. With the proper sync, the display block with its 2000 character blocks will all be lit at the correct places on the screen.

Besides the six signals to produce the RGBI picture, CN10 also outputs another separate signal called Monochrome. This is also an 80-column signal. It is a composite monochrome TV signal that is derived from the R, G, B, I, horizontal sync and vertical sync developed in the 8563. It emerges from

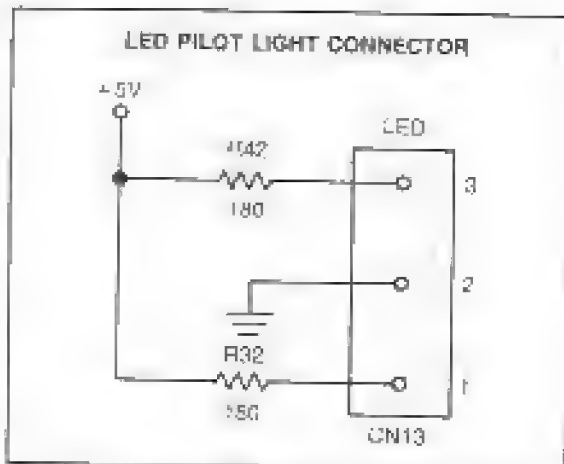


Fig. 23-17. Connector CN13 is the little 3-pin LED pilot light connector.

pin 7 and can be used in a composite TV monitor. This signal is quite like the 40-column composite TV signal that CN8 outputs from pin 4, except that there is no color component.

Pins 1 and 2 of the RGBI connector are grounded. That was the final port connector. CN111 and CN12 are the inputs for the power supply ac adapter box. That is covered in the next chapter. Figure 23-17 is the LED pilot light connector.

24. The Power Supply

The power supply in the C128 is shown in Fig. 24-1, Fig. 24-2 and Fig. 24-3. The supply is found in the ac adapter box, Fig. 24-4, and also on the 128 printboard. The ac adapter, which is simply a giant version of a calculator ac adapter, is in the box accompanying your 128. It has two main sections. First is the power transformer part, Fig. 24-1. In one leg of the transformer secondary is a 1.6 amp fuse, F1, and the other leg is the 4 amp fuse, F3. Fuse F1 is in the winding that supplies a 9.6 Vac to pins 3 and 5 to CN11 on the side of the computer case. Fuse F3 protects an 18.7 volt dc output to the other circuits in the power box.

The computer needs the 9.6 Vac and three dc voltages to operate. The 9.6 Vac is injected directly to CN11 and then into the computer. The 18.7 Vdc is sent to a power regulator circuit, Fig. 24-2, in the power box. The regulator circuit converts the 18.7 Vdc to a well regulated, heavy duty, +5 Vdc that

powers practically every chip in the machine. The regulator injects the +5 volts into pin 1 of CN11.

Pins 2 and 4 of CN11, Fig. 24-3, are grounded. Pin 2's grounding is accompanied by shielded wire and two .01 capacitors to ground. This grounding is inside the 128 case and is there to make sure no spurious oscillations take place and hurt the data processing.

The two 9 Vac inputs to the main board are out of phase with each other and thus form a potential of 18 volts ac. This voltage is processed and two source voltages produced. One is an 11.5 volt dc and the other an 11.9 volt dc source.

Power supply voltages are among the most common in the computer. Typically, the machine simply goes dead if the source voltages are not applied. The fuses could blow, rectifiers short, regulators open up, the transformer or another component could start smoking, and other things.

POWER TRANSFORMER

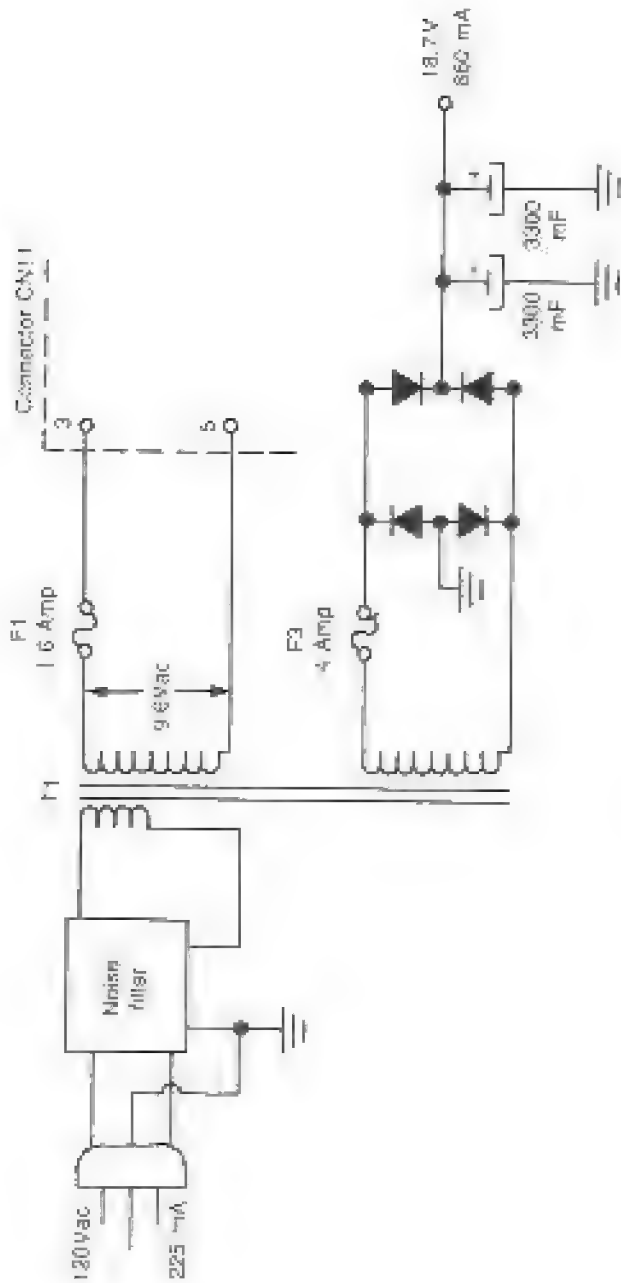


Fig. 24-1. The power transformer and fuses are located in the ac adapter power box. The transformer outputs 9Vac to CN11—the power supply input plug on the main board. The transformer also sends 18.7 volts dc to another section of the power app.

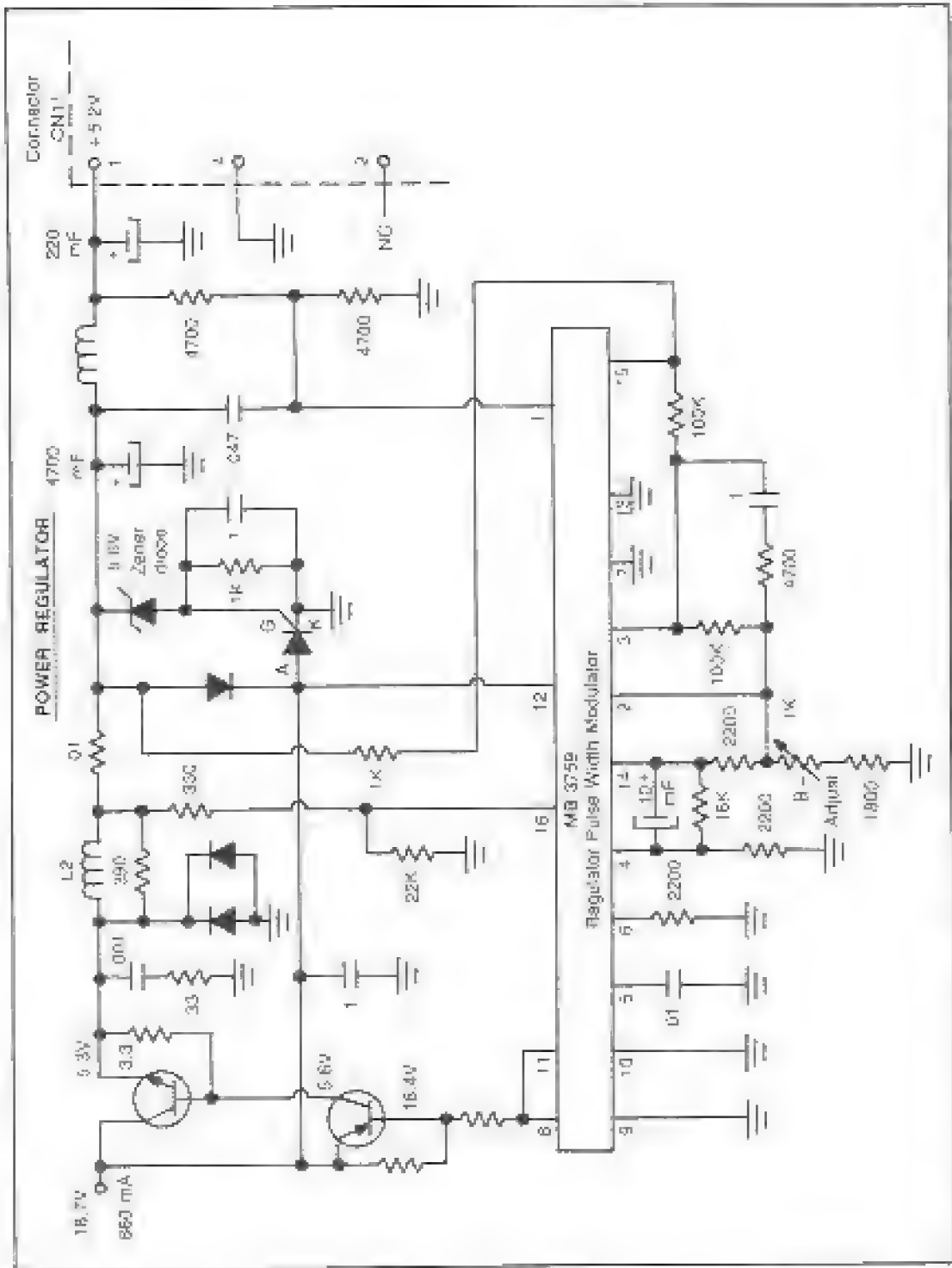


Fig. 24-2. The 18.7 Vdc is input to the power regulator section of the power box. The regulator circuit transforms the 18.7Vdc to a slightly regulated +5.2 volt dc that is input to pins 1 of CN11.

POWER SUPPLY ON MAIN BOARD

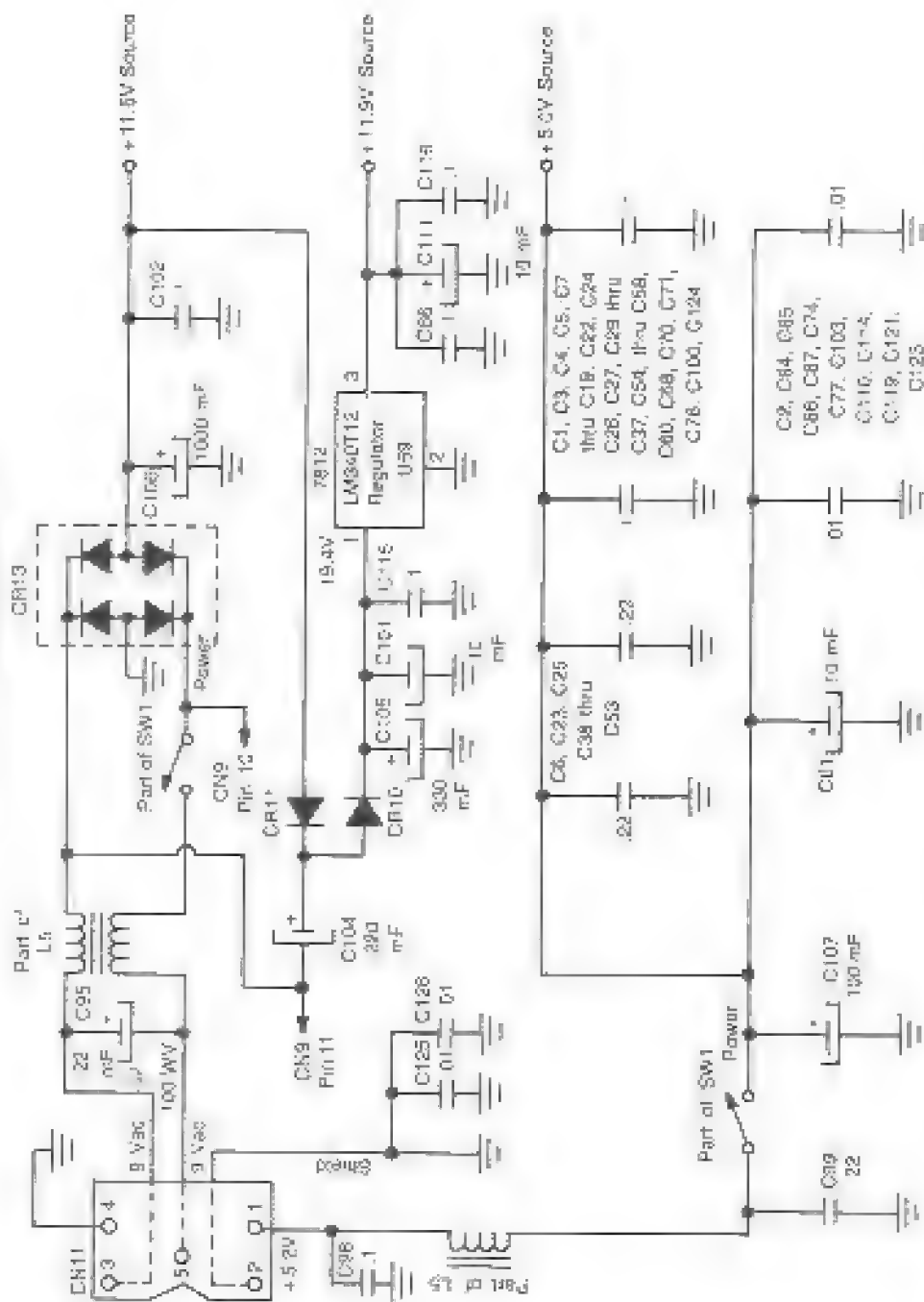


Fig. 24-9. Connector CN11 is wired to the power supply section on the main board. This circuit outputs four voltages to the C128; three dc and one ac.

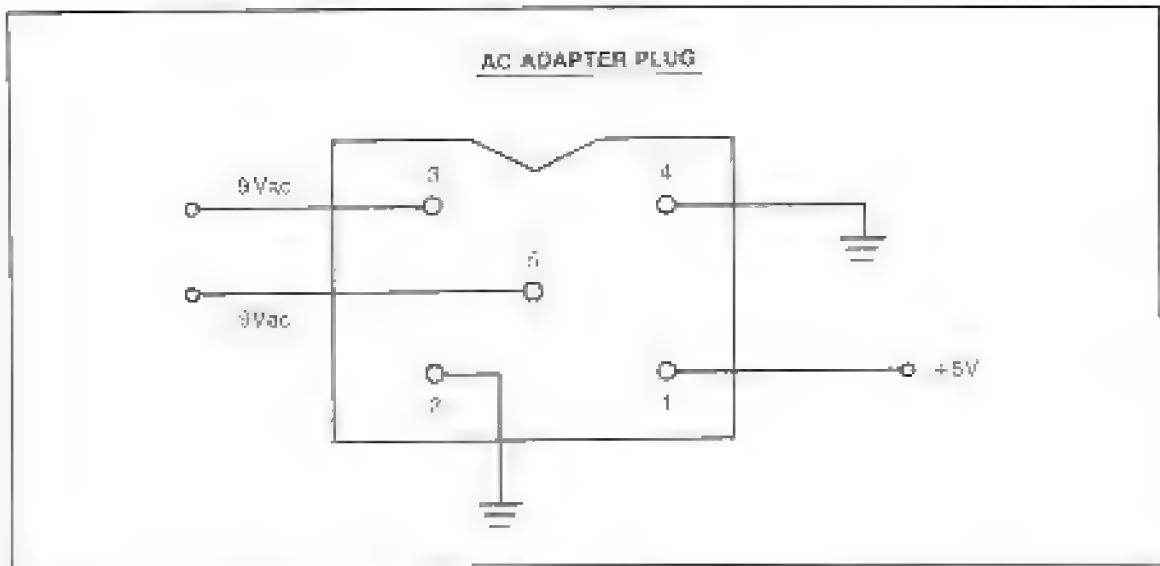


Fig. 24-4 This is the ac adapter plug coming off of my power box and connecting to CN11. Other versions of this power box exist.

Aside from the symptoms of smoking, which provides instant identification of the trouble and perhaps even the bad component, the best way to check out a defective power supply is step-by-step.

FIRST STEP

When the computer goes dead and you suspect power supply trouble, the first question is "which power section has failed, the ac adapter or the print-board part?"

Without plugging the ac adapter into CN11 in the C128, plug the adapter into the 120 Vac house wall socket. No off-on switch is in the power box so it comes on when plugged in.

A number of different ac adapters come packaged with the C128. Some are potted in epoxy. Others are easy to look at and test. Whichever one you own, they all output the same voltages. If you test their outputs then you should obtain the following results if the circuits in the box are okay.

Figure 24-4 shows the layout of the plug that mates into CN11. With the keyway at 12 o'clock, counting clockwise: pin 4 is at 2 o'clock; pin 1 at 5 o'clock; pin 2 at 7 o'clock; pin 3 at 10 o'clock; pin 5 is the center of them all. With a vom, place the positive probe on pin 1 and the negative on pin 4.

Pin 4 is ground and pin 1 is a +5 volt output. The vom should read +5.2 volts or pretty close to it if the voltage is okay. If the +5 volts is missing, then the next step is covered in the following section. Pin 2 can be ignored—it has no connection.

When the +5 volts is present, check the voltage across pins 3 and 5. You will be testing for an ac voltage. About 9 or 10 volts ac should be present. The steps to take when it is missing are covered in the next sections.

Fuse Considerations

When the +5 volts is gone, that explains the reason for the dead computer. If your power box is the type that has the circuits encased in epoxy, then it has been recommended that it is more trouble than it is worth to pull and hack the circuits free. A new power box costs about \$40 and is probably the easiest and most practical solution.

If your power box is easy to open and the circuits fairly accessible, then you'll see the fuses. Different boxes have different size fuses but they run between 1 and 4 amps in ratings. Test the fuses and if you find a bad one, change it with the same value. Do not overfuse as that could be dangerous.

If the +5 volts is missing then the fuse is in the

transformer secondary winding that produces the 18.7 volts at 860 mA. It could be a 3 or 4 amp fuse. Figure 24-1 shows the secondary circuit for the +5 volts.

Should the 9 Vac be missing, there is another fuse in the other transformer secondary winding. If it opens, then the 9 Vac will disappear. Try replacing that fuse. It could be a smaller value fuse of perhaps 1 or 2 amps.

Chances are good that if you find a bad fuse and change it that the computer could start operating again. However, if it does not and the fuse blows again, there is a short in the power input circuits and they must be checked out and cleared before the fuse will stop blowing. Let's go inside the power supply and see how it works, so we can puzzle out how it will fail. A typical C128 power box has two sections: the power transformer section and the power regulator.

Power Transformer

Figure 24-1 starts with the ac three-pronged polarized plug at the left. The box circuits are designed to draw about 225 mA at 120 Vac. Note the center line is grounded. The two live lines enter a noise filter. The filter is simply some capacitors across the three lines. Typically, the capacitors are .01's with a high working voltage such as 1000 working volts. These capacitors are there to smooth out any high frequency noise pulses that might enter the box with the ac line input voltage.

The 120 volt ac enters the primary of T1, the power transformer. The two secondary windings are step-down types. The top winding has the 120 Vac stepped down to 9.6 Vac. The bottom winding has the voltage stepped down to about 20 Vac. The top winding is connected directly to pins 3 and 5 of CN11. The 9 Vac gains entrance to the printboard in that way. The bottom winding places the 20 Vac into a bridge rectifier network.

The four diodes that make up the bridge rectifier change the ac input into a pulsating dc output. There are two legs in the bridge, each with two diodes in series. The bridge rectifier acts as a full-wave rectifier without the need for a center-tapped transformer.

The output of the bridge is sent into a pair of 3300 mF filter capacitors. The capacitors change the pulsating dc into a smooth dc with a resultant voltage of 18.7. This voltage can supply up to 860 mA of current. The 18.7 Vdc is then sent to the second section of the power box, the Power Regulator.

Power Regulator

The 18.7 Vdc is going to finally result in +5 volts after it is processed in the regulator circuit. In the example circuit I'm using as an illustration, there are a lot of components that are going to work over the 18.7 Vdc. Basically what they must do is convert the 18.7 Vdc into a specific, stable +5 volts, and hold that +5 volt level over wide variations of the input +18.7 volts or wide variations in the load the +5 volts must supply. In other words, the voltage must be strictly regulated. If the input line voltage should drop somewhat as sometimes happens, or some cartridge is plugged into the Expansion Port and draws more than normal currents, then the regulator must hold that +5 volts steady.

What that means is, the circuits are going to sample the output voltage and if it is not precisely the desired +5.2 volts, the error will be noted, a correction voltage generated and the amount of voltage in error will be erased. This results in keeping the voltage at +5.2 volts no matter what.

Without going into a long dissertation on regulator theory, the idea is that the Power Regulator system receives the +18.7 volts and converts it to a +5.2 volts that will work in a trouble-free manner in the sensitive digital circuits. The +5.2 volts is injected into CN11 at pin 1. In Fig. 24-3 I show pins 4 and 2 as being grounded. It is possible that some power boxes could have these pins reversed and either or both could be grounded.

When you test the ac adapter and find the +5 volts missing, and the 9 Vac is present, chances are good that the regulator circuit has failed. If you have an epoxy encased power supply, then the easiest way out is to purchase a new reliable power box as mentioned earlier.

the easiest way out is to purchase a new reliable power box as mentioned earlier.

Should you be determined to try to fix it, here

is the procedure. The box is semisealed by the manufacturer. It is possible to take it apart but chances of ruining it are good. The first step is, do not try taking it apart while it is plugged in! It does connect to 120 Vac which is a dangerous voltage when plugged in.

With a thin screwdriver you could gingerly pry the case apart. You will then encounter a thick epoxy layer. You will have to break off the layer, being careful since the epoxy contains embedded wiring. The components will be buried beneath the epoxy.

Should you be able to safely repair the wiring faults, retrace your steps until you have the unit back together. Again let me mention that the epoxy encased unit was designed as a throwaway and it might not be practical or reliable to repair it unless you are an experienced wireman.

If you own one of the power boxes that comes apart easily, as shown in Fig. 24-4, then you can attempt repairs and if you can't fix it, a Commodore repair shop can.

Power Supply on Main Board

Figure 24-3 has CN11 in the upper left hand corner. The regulated +5 volts enters pin 1 and the 9 Vac comes in through pins 3 and 5. The +5 volt input is bypassed with C98 a .1 capacitor. It continues through L5 to the power switch, SW1. When SW1 is turned on, the voltage courses on over some filters and a host of small capacitors such as .22's, .01's, and .1's. It enters and powers all the +5 volt points. Note the input +5.2 volts drops to +5.0 volts due to wire and coil resistance. The +5 volts, of course, must remain well regulated at +5 volts.

The 9 Vac enters at pins 3 and 5 of CN11 and passes around a .22 mF filter and through L5. The line is also switched by part of SW1. After L5 the ac is inserted into the bridge rectifier CR13.

The bridge changed the 9 Vac to a pulsating dc. The dc is then filtered to a clean but unregulated dc by the 1000 mF filter. The resultant voltage is a +11.5 volts. This voltage is not going to be used to power chips so strict regulation is not needed.

A tap is made at the +11.5 Source point. This tap goes through a pair of series diodes and then

into a 7812 regulator, U59 on the printboard. This regulator works like the more complex circuit in the power box but is not as critical.

U59 is a good test point on the printboard. Its output is +11.9 volts at pin 3. If that voltage is present, the entire 9 Vac line is shown to be working okay. Both sides, input at pin 1 and output at pin 3 of U59 are heavily filtered.

Note that a tap is made of the 9 Vac to the right of L5. This line goes to pin 11 of CN9, the User Port. In addition to the left of CR13 is a common 9 Vac tap that goes to pin 10 of CN9. This voltage is used to power the Time Of Day pins of the two CIAs. This ac is the power company's 60 Hz frequency used to time all electric clocks.

POINT BY POINT CHECKOUT

When the computer is dead and the fuse and on-off switch are intact, the vom is needed. First stop is at the bridge rectifier, CR13, inputs on the printboard. There should be 9 Vac present. If the ac voltage is missing, then work back across the components to pins 3 and 5 of CN11. The components are L5 and C95. An open winding in L5, a shorted power switch, C95 or a board short could kill the correct voltage. When the 9 Vac is present at the input of CR13, check the output. The 9 Vac is rectified and filtered to the +11.5 volts. If it is missing, then CR13 is probably defective. If it is present, then the line checks out okay and you move on to the next leg.

Check U59 pin 1. A +19.4 volts should be there. If it is not, then the two diodes, CR11 and CR10, and the filters C104, C105, C101 and C115 become suspects. Test them for faults. When the voltage on pin 1 is correct, move your touchdown point to pin 3. About +12 volts should be there. I show the designed voltage of +11.9 volts in Fig. 24-3. If the voltage is missing there, then the prime suspect becomes U59. Another way the voltage could be missing is if one of the three filter capacitors is shorted. They are C86, C111 and C116. If one of the them opens or develops a high resistance short the voltage could be changed from +12 volts to some other value. Test the capacitors if there is a missing or wrong voltage and U59 proves okay.

When those two legs check out okay and the +5 volts is missing on the board, first stop is pin 1 of CN11. If the +5 volts is at CN11, then all the components in the +5-volt input line from CN11 to the chip inputs are suspect. The input line consists of a series coil L5, the series off-on switch, part of SW1 and all the bypass and filter capacitors shown in the line. Especially vulnerable are C107, a 100 mF filter, and C61 a 10 mF. If L5 should open or any of the capacitors short, the +5 volts could disappear.

Starting at pin 1 of CN11, where the voltage is present, move down the line and take voltage readings of the series components. As soon as you pass a component and the voltage disappears you have just crossed over the defect. Don't forget the off-on switch SW1. It has two sections; one in the upper leg and the second in the bottom leg. One section could break open while the other side remains intact.

When the +5 volts is missing at CN11 pin 1, the trouble is indicated to be in the power box. The first stop is at the collector, C, of the 5 volt regulator npn transistor. There should be +18.7 volts there. If the voltage is missing, then the fuse F3 becomes suspect. If the fuse is blown then the bridge rectifiers have probably shorted and need replacement. A less likely but possible trouble is a shorted 3300 mF filter in that line.

Another possibility is the 5 volt regulator transistor itself. If it shorts, it could kill the +18.7 volt input. Remote possibilities are the 5.6 volt Zener diode, the silicon controlled rectifier, SCR, the power transformer T1 and the input power line cord.

When the voltage is present at the 5 volt regulator transistor collector, check the voltage at the emitter, E. There should be about +5.3 volts there. If it is missing, then both of the transistors become suspects. Either one or both of them could be defective.

Should the transistors test out okay, the prime suspect then becomes MB3759, the Regulator Pulse Width Modulator. The correct voltages are shown

Table 24-1. MB3759 DC Voltages.

Regulator Pulse Width Modulator MB3759 DC Voltages	
PIN	DC Voltage
1	2.8
2	2.6
3	3.0
4	0.5
5	1.6
6	3.6
7	
8	13.3
9	—
10	—
11	13.3
12	18.7
13	
14	5.1
15	5.2
16	5.2

on the schematic. If they check out incorrectly, then first test the components off the pin with the wrong voltage on it. If the components are okay, then chances are good that the MB3759 is defective and needs replacement (see Table 24-1).

Should all of the above test okay and the voltage is correct on the emitter of the regulator transistor, then the rest of the components between the emitter and the output pin to pin 1 of CN11 become suspect. If L2 or L3 should open up, this condition would occur. If one or both of the two diodes beneath L2 should short, it would kill the voltage. Should the 4700 mF or 220 mF filter short, that occurrence would make the voltage disappear.

Power supply troubles are the most common that happen to computers. They are also the easiest to fix. The ac and dc voltages can be tested from the input pins to any destination. If you start at an input pin and the voltage is there, then you can check out the entire line by following it and crossing over component by component. If the voltage suddenly disappears, then you have just passed over the defect.

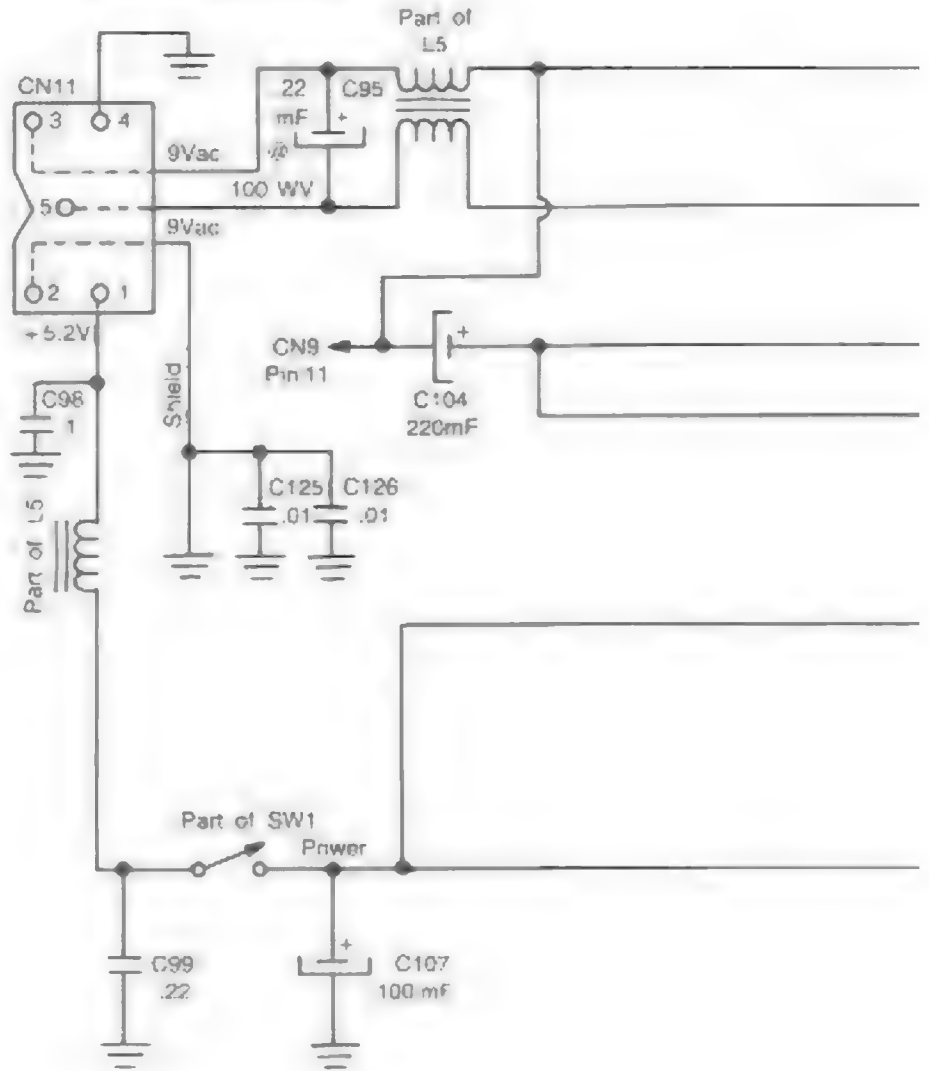
Appendix

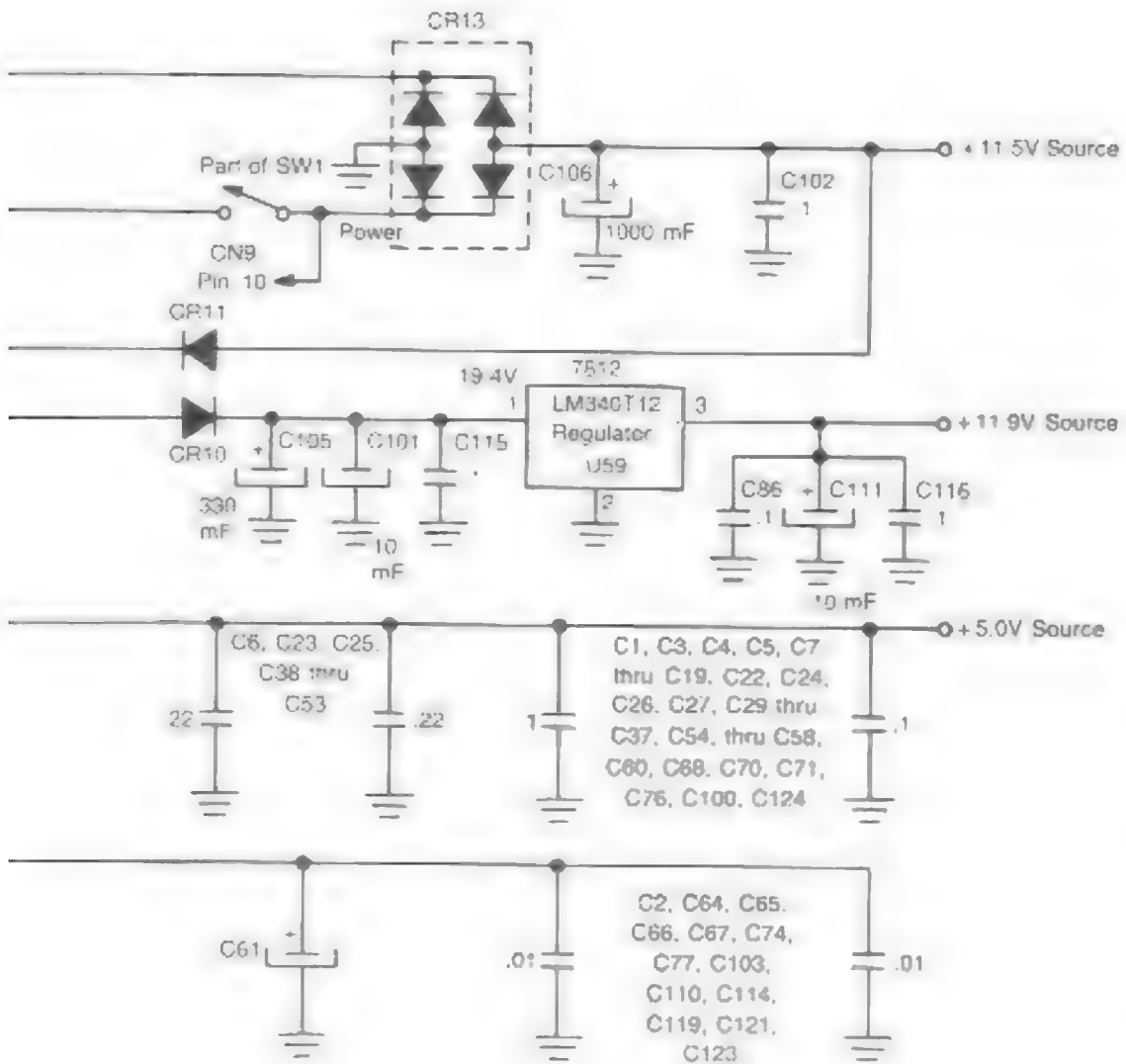
TROUBLESHOOTING & REPAIRING THE C126 Main Board (11 sheets)

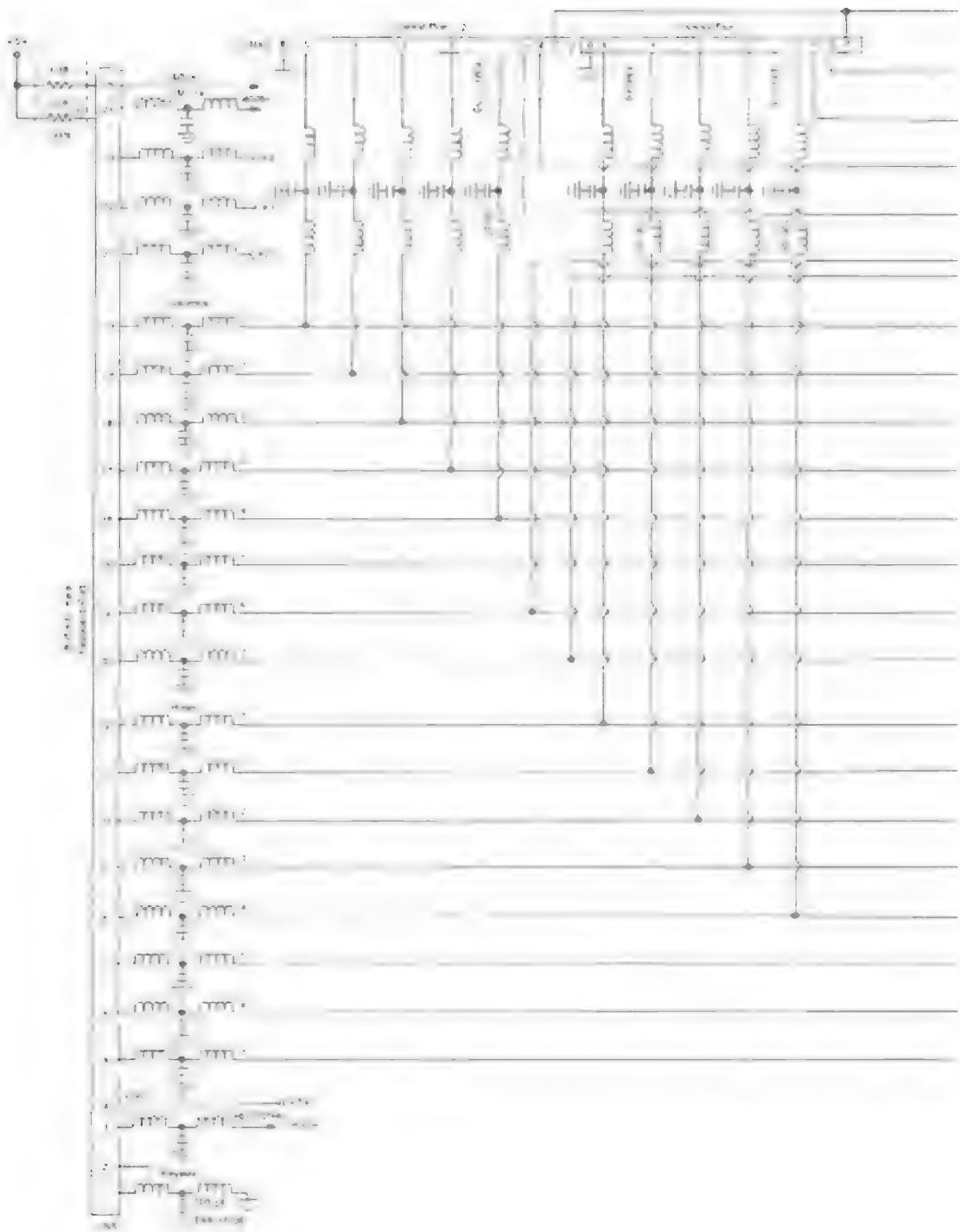
U Number	Generic Number	Given Name	Master Schematic
U1	6526	Complex Interface Adapter	2
U2	4086B	Quad Bilateral Switch	2
U3	74LS138	1-of-8 Decoder	11
U4	6526	Complex Interface Adapter	8
U5	6581	Sound Interface Chip	11
U6	9502	Microprocessor	3
U7	8722	Memory Management Unit	4
U8	74LS08	Quad 2-Input AND Gate	5, 7
U9	74F32	Quad 2-Input OR Gate	6, 7
U10	Z80	Microprocessor	3
U11	8721	Programmable Logic Array	4
U12	74LS373	Octal 3-State D Latch	3
U13	74LS244	Octal 3-State Driver	3
U14	74LS257	Quad 2-Input Multiplexer	11
U15	74LS257	Quad 2-Input Multiplexer	11
U16	74LS14	Hex Schmitt Trigger	7, 8
U17	74LS373	Octal 3-State D latch	3
U18	390059-01	Character ROM	9
U19	2018	Color RAM	9
U20	4066B	Quad Bilateral Switch	9
U21	8564	Video Interface Chip	9

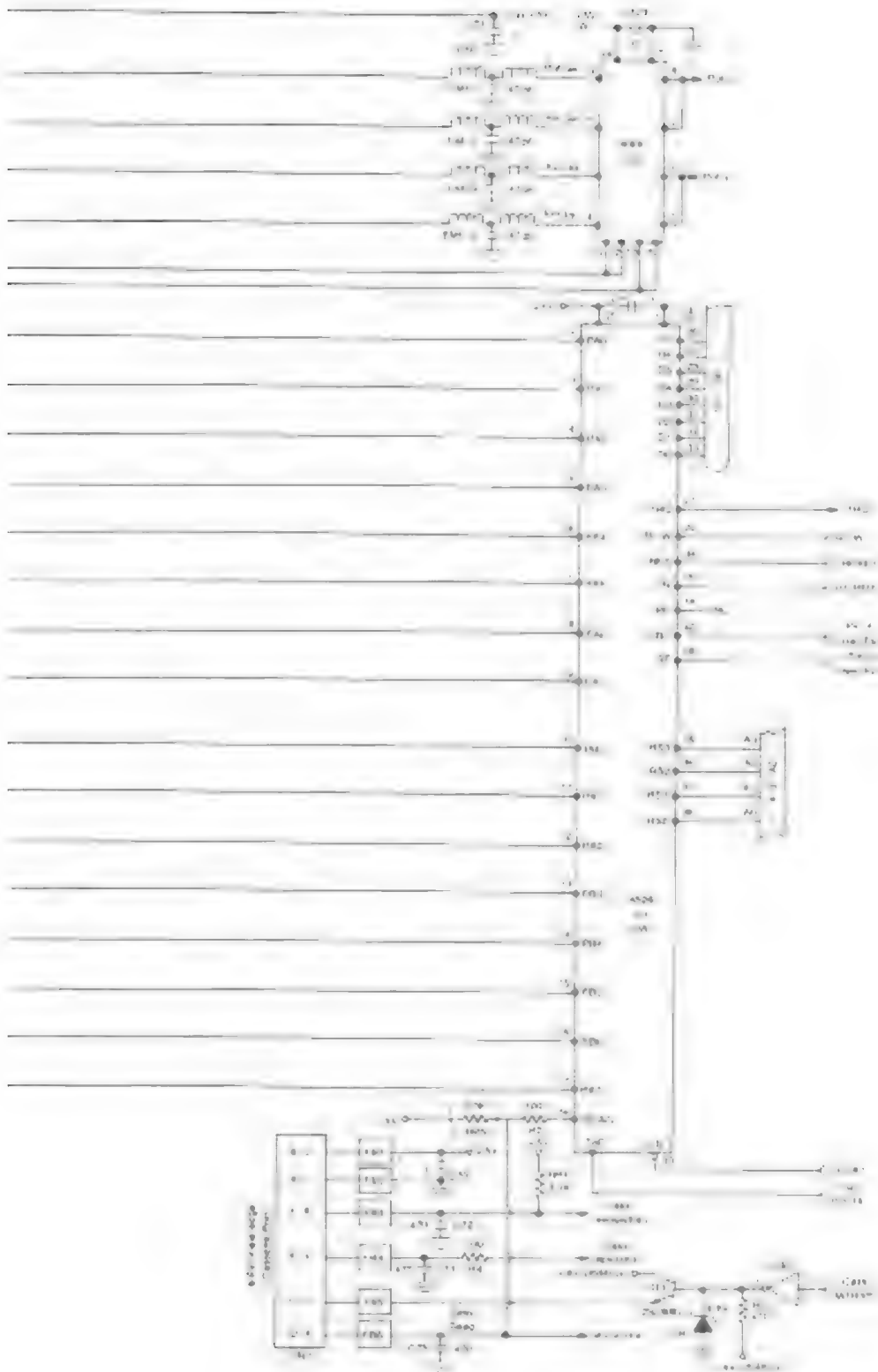
U22	6583	Video Controller	10
U23	4418	DRAM	10
U24	74LS244	Octal 3-State Driver	10
U25	4416	DRAM	10
U26	74LS257	Quad 2-Input Multiplexer	9
U27	3856	Timer	7
U28	8701	Clock	9
U29	7406	Hex Inverter Buffer	7, 9, 10, 11
U30	7406	Hex Inverter Buffer	2, 7, 8
U31	74LS00	Quad 2-Input NAND Gate	11
U32	251913-01	Read Only Memory	5
U33	316018-02	Read Only Memory	5
U34	316019-02	Read Only Memory	5
U35	316020-03	Read Only Memory	5
U36		Empty Socket for Functional ROM	5
U37	7406	Hex Inverter Buffer	7
U38-U39	4154	DRAM	8
U54	74LS32	Quad 2-Input OR Gate	4, 10
U55	74F245	Transceiver	9
U56	74LS74	Dual D Flip-Flop	9
U57	7407	Hex Buffer	8, 10
U58	74LS03	Quad 2-Input NAND Gate	7
U59	7812	12 Volt Regulator	1
U60	7407	Hex Buffer	7
U61	74LS08	Quad 2-Input AND Gate	3, 4
U62	74LS244	Octal 3-State Driver	8
U63	7406	Hex Inverter Buffer	7, 8

POWER SUPPLY ON MAIN BOARD



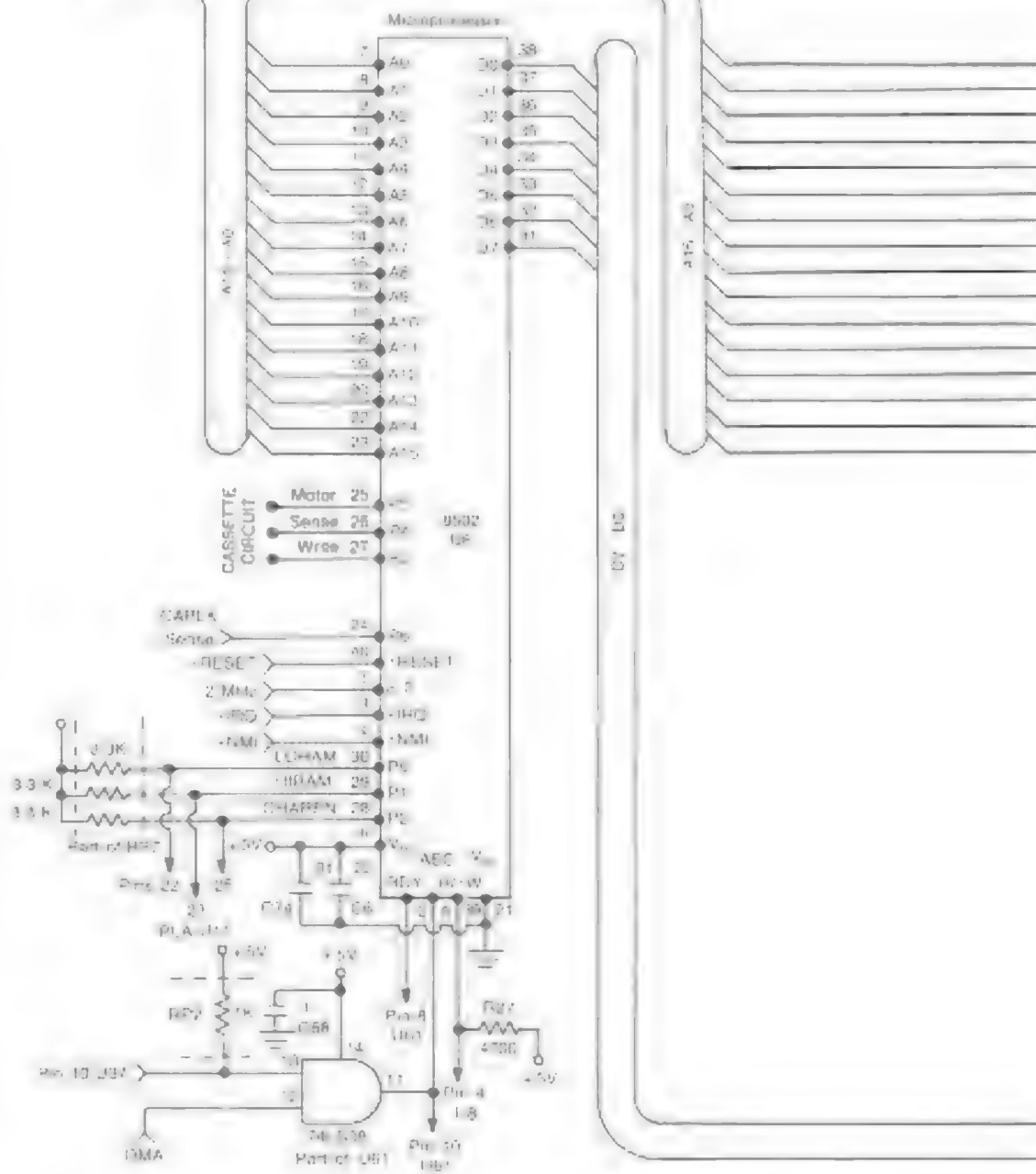


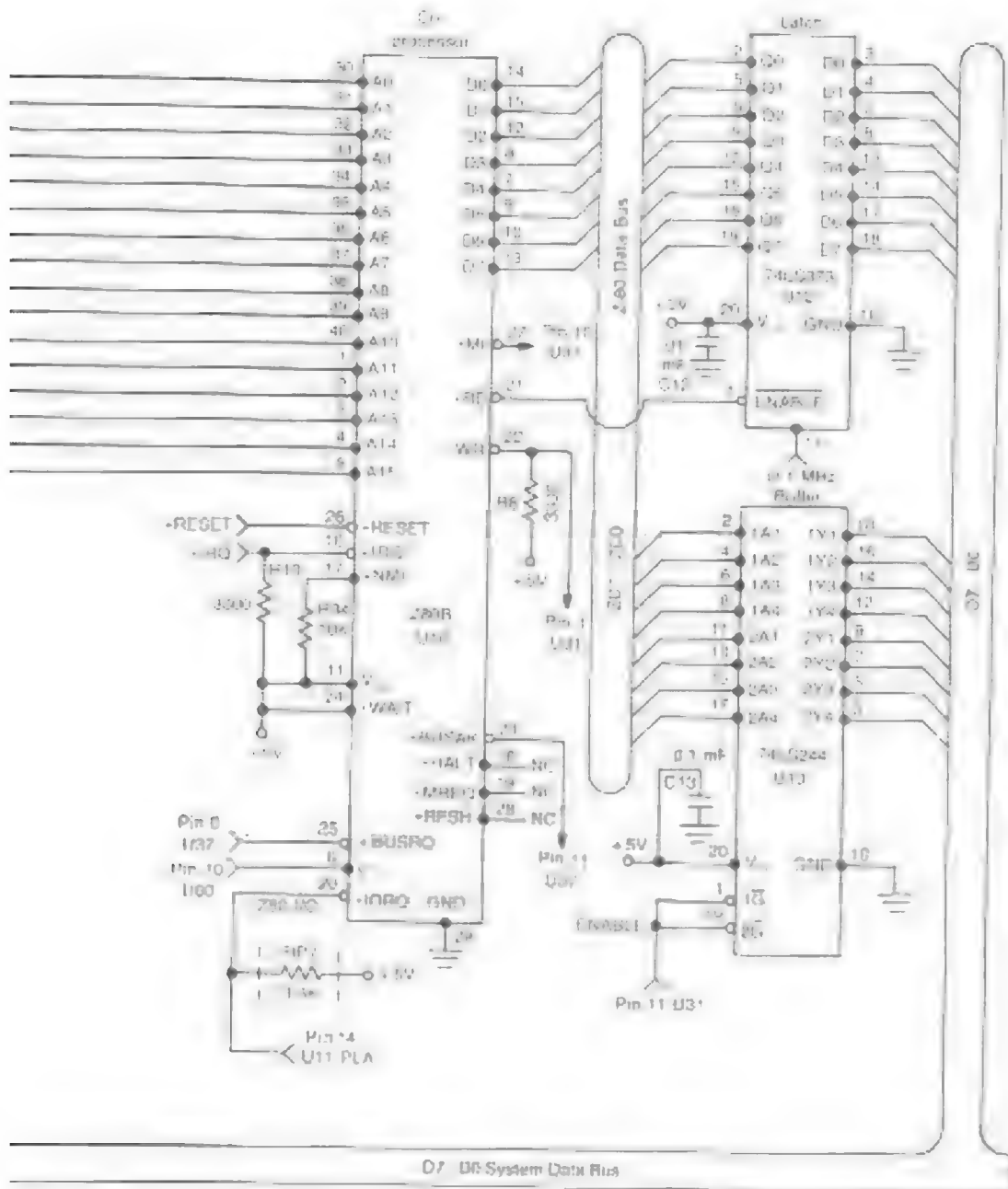




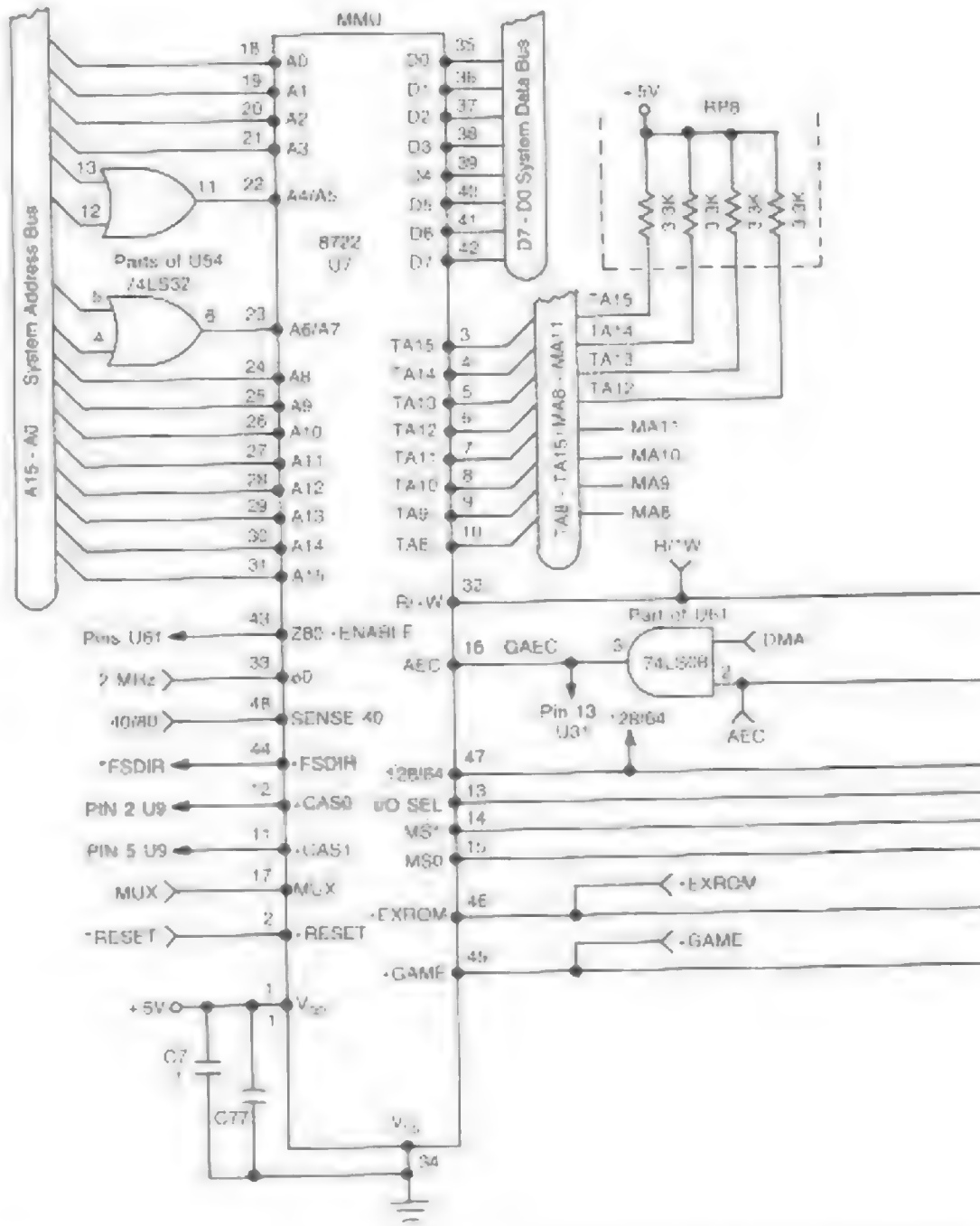
MICROPROCESSORS

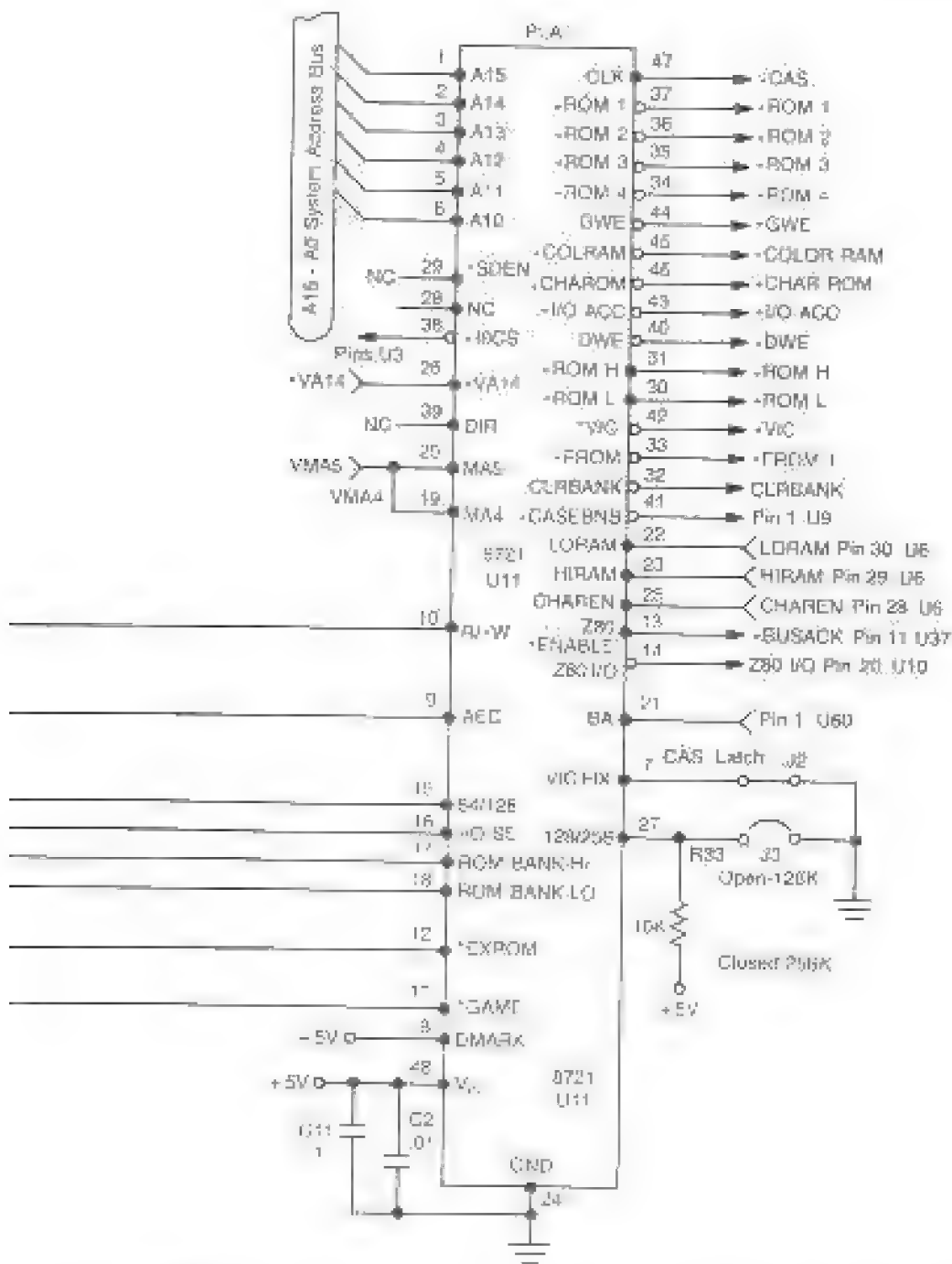
A15 - A0 System Address Bus

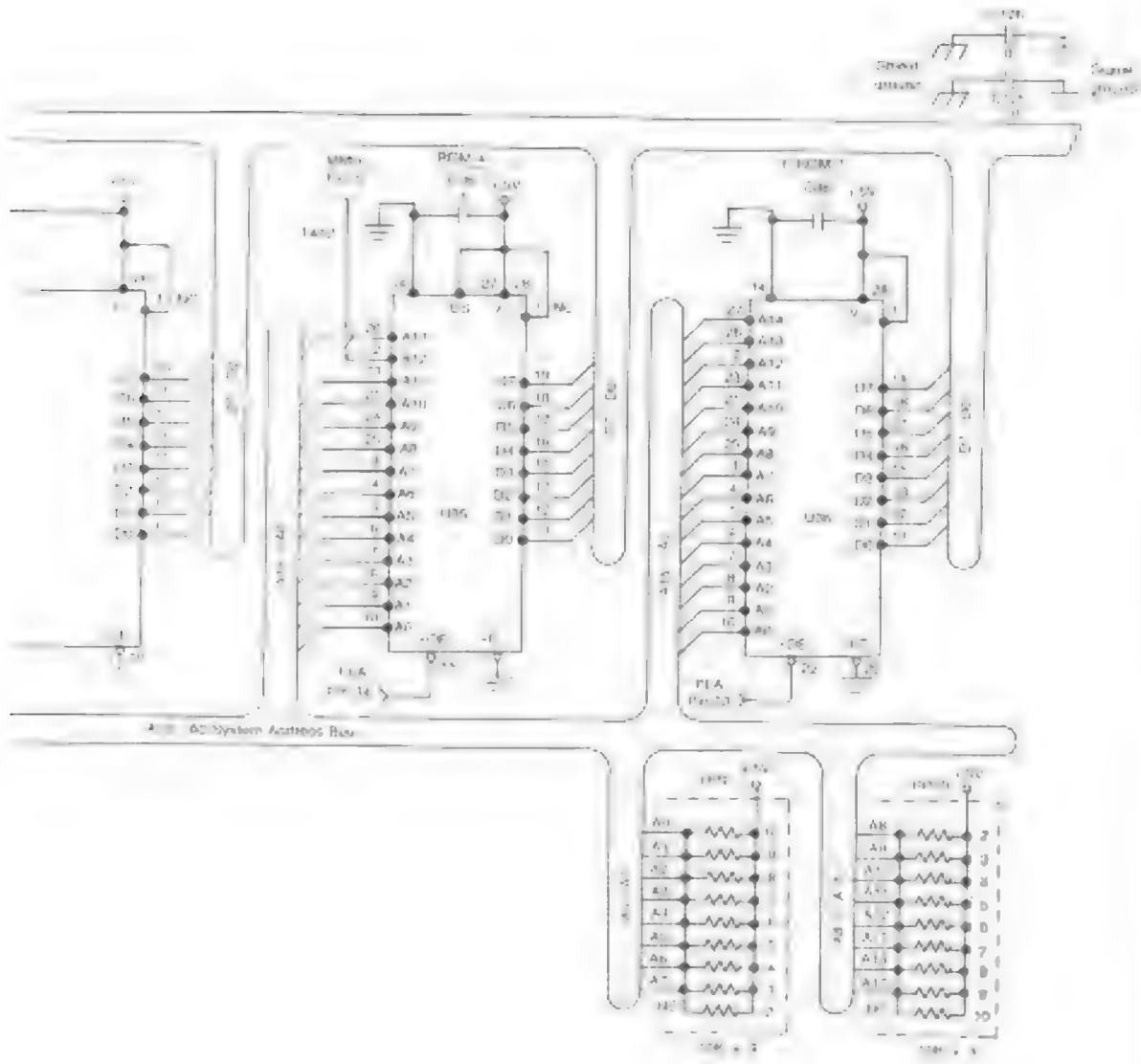


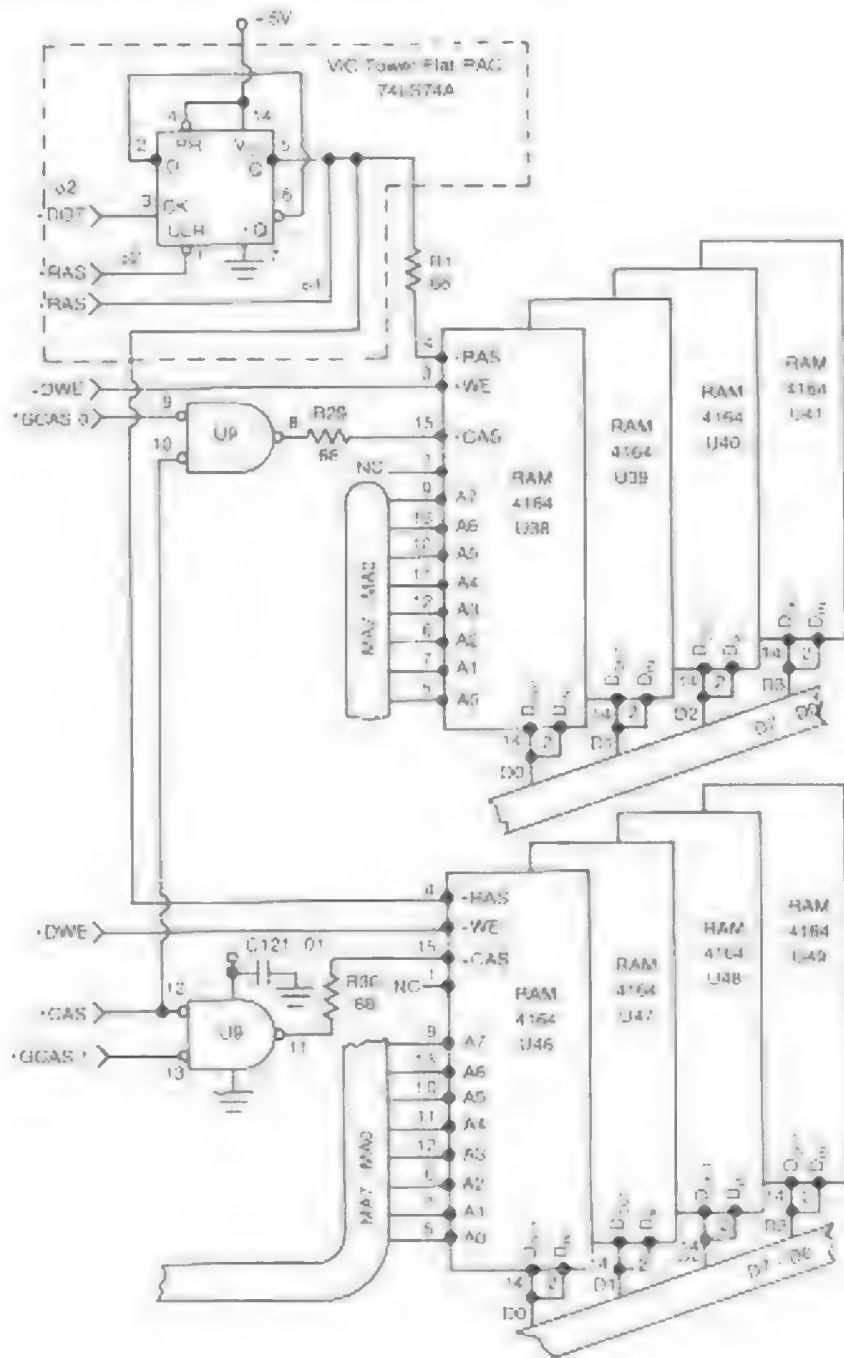


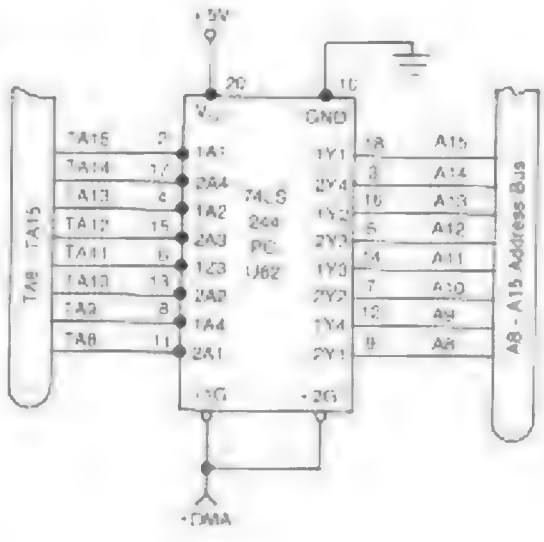
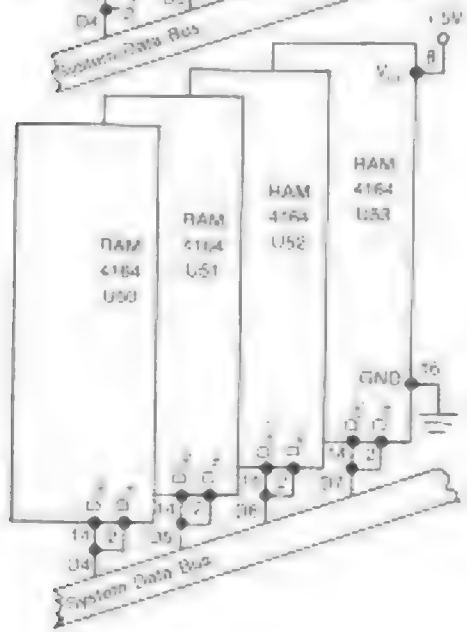
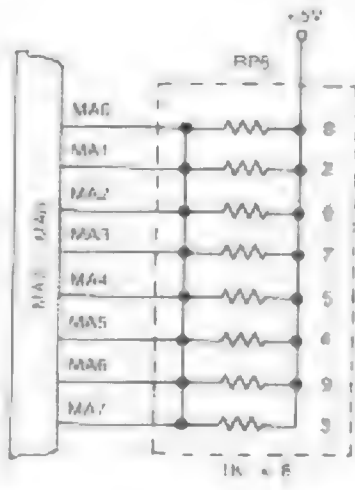
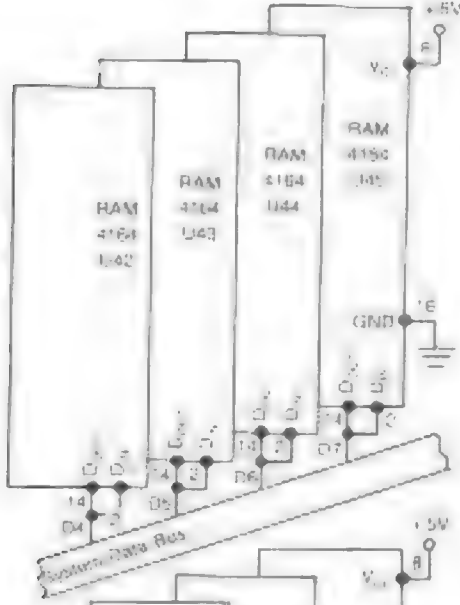
D7 DR System Data Bus

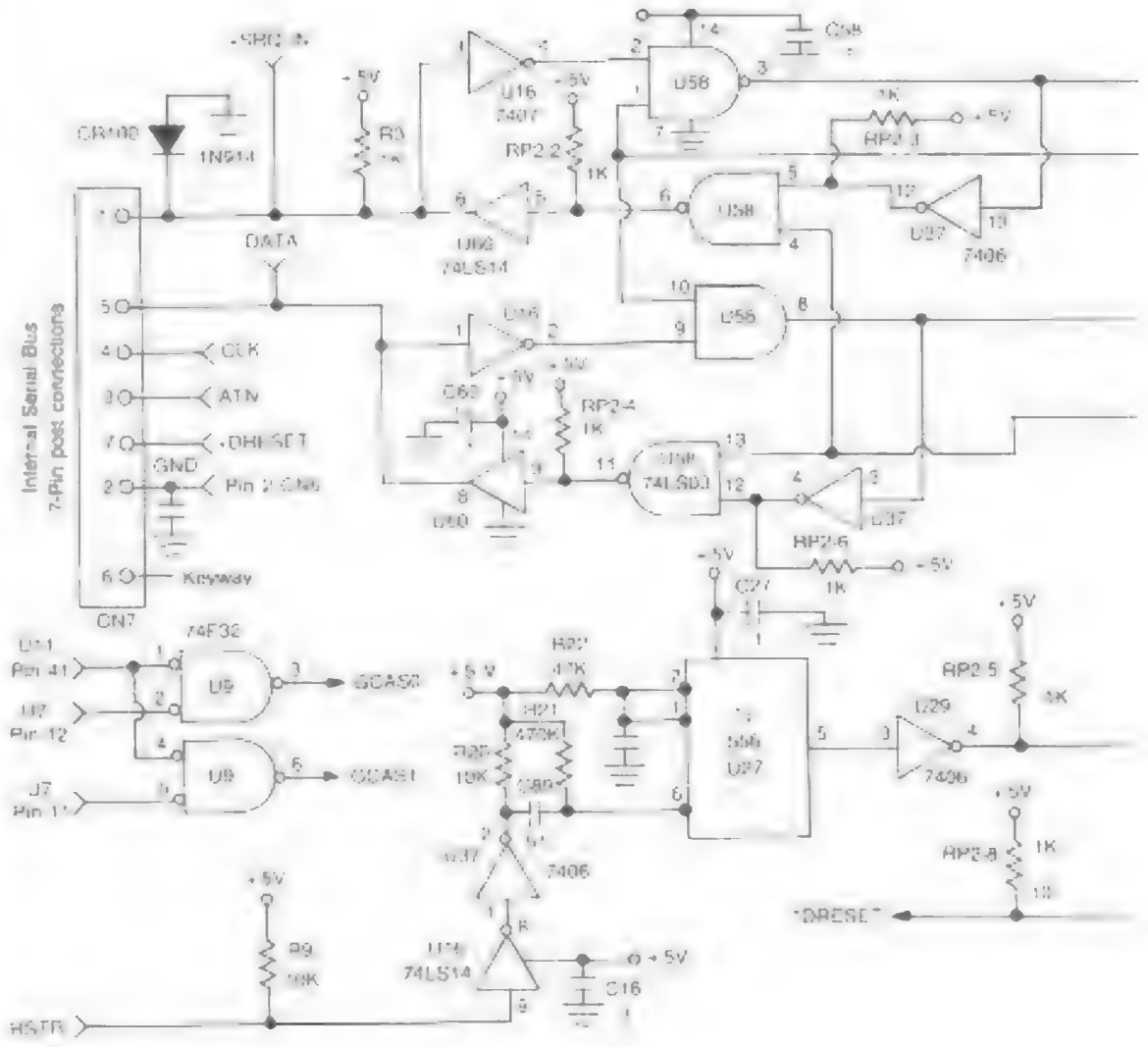


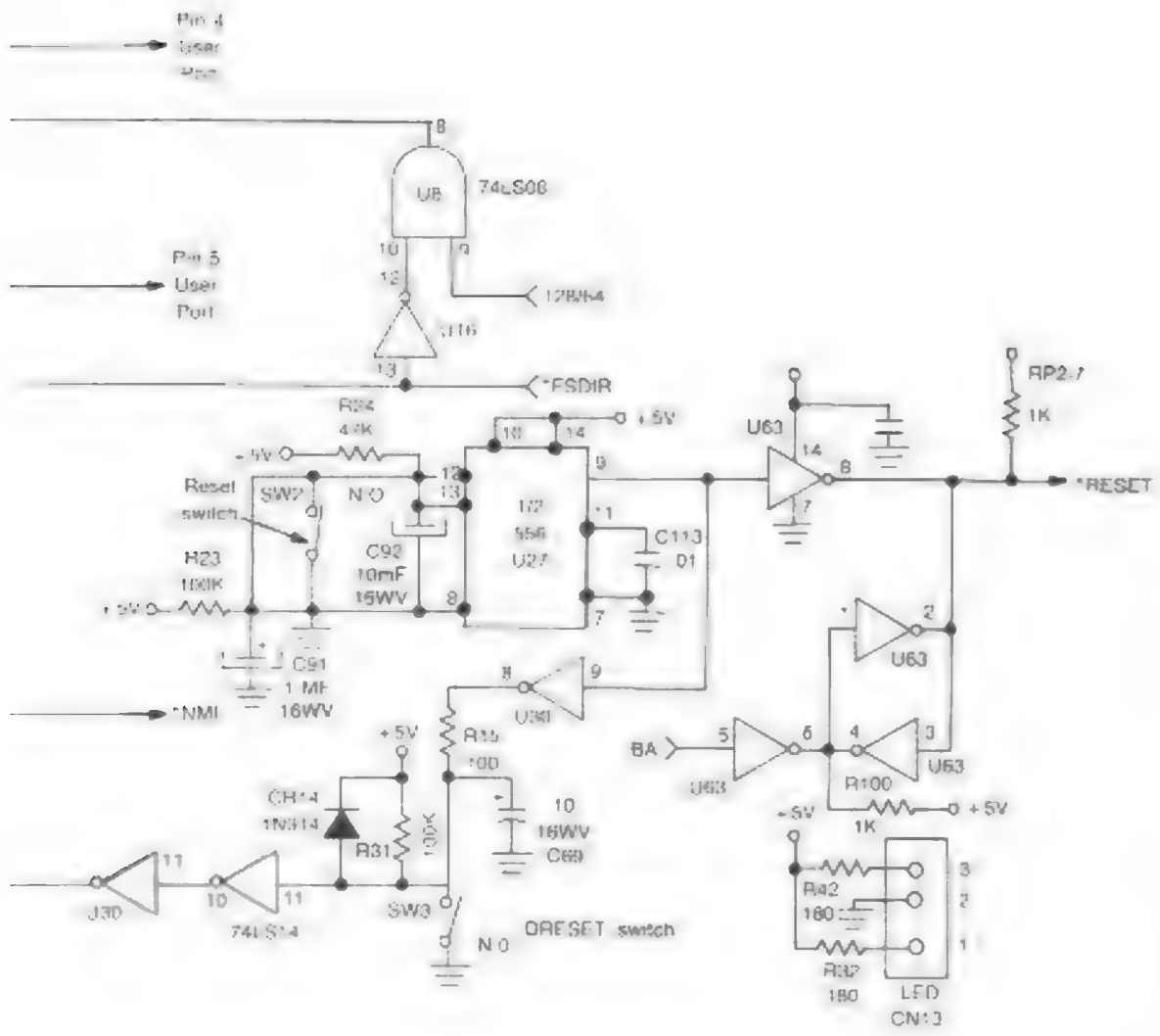


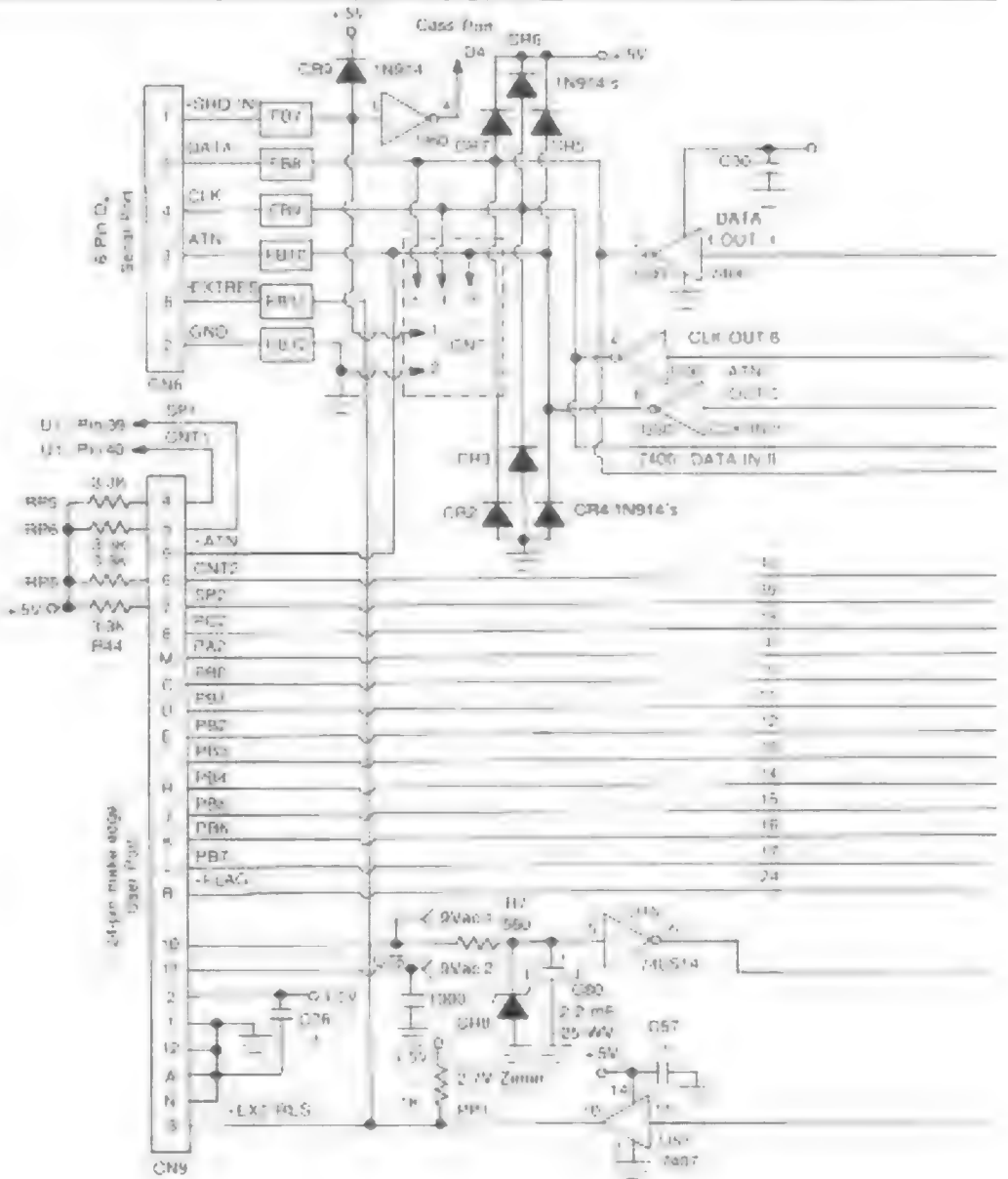


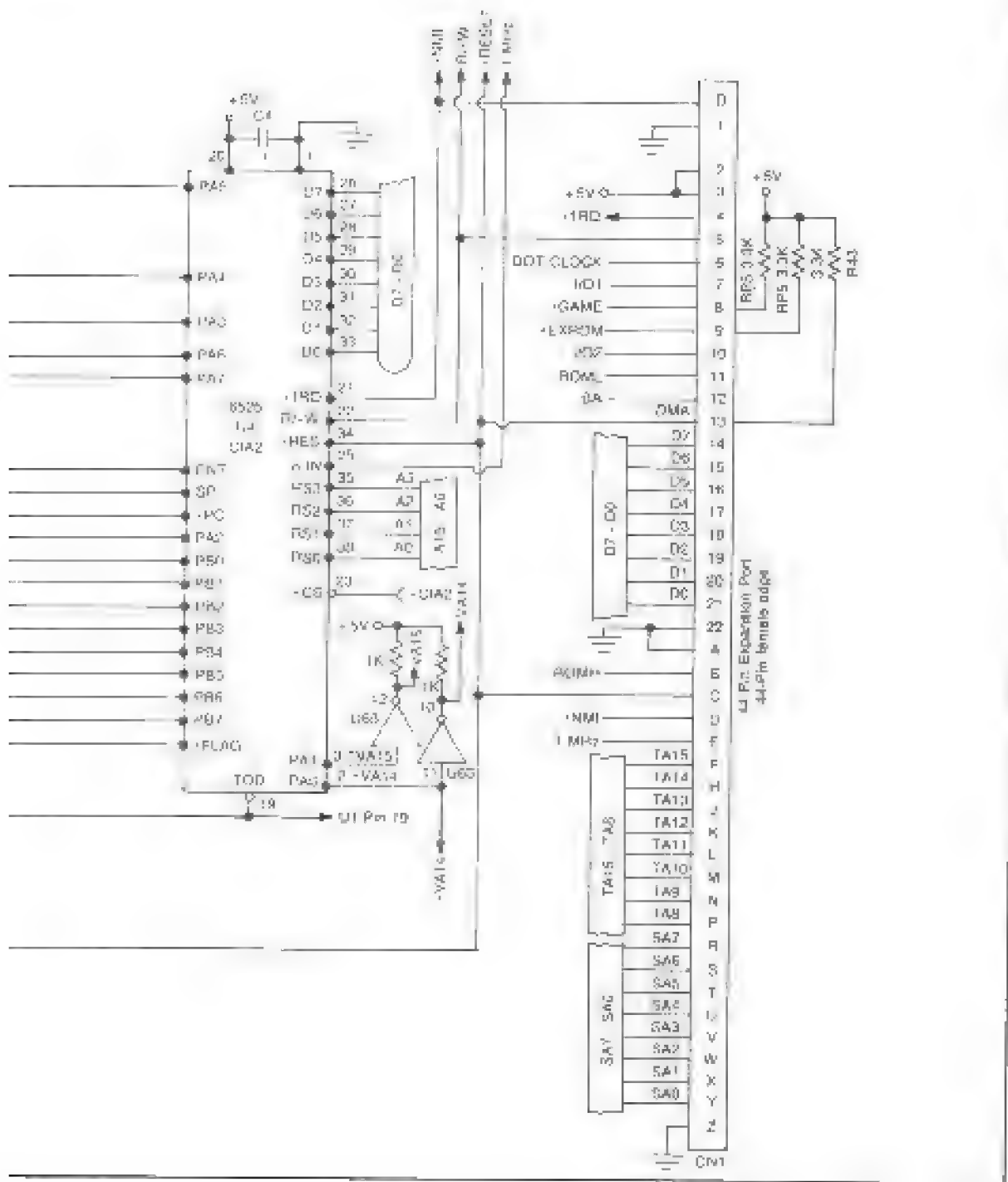


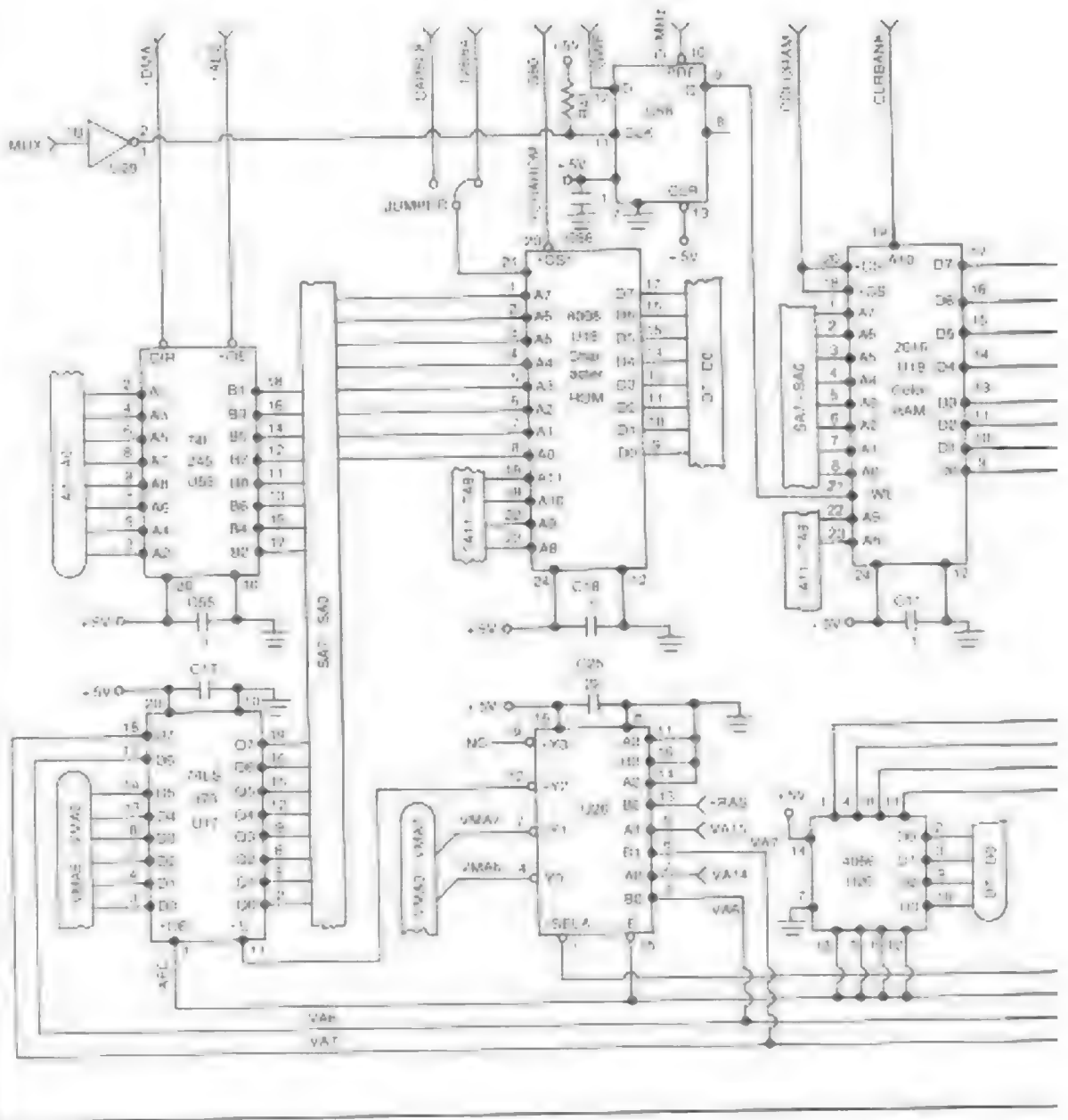


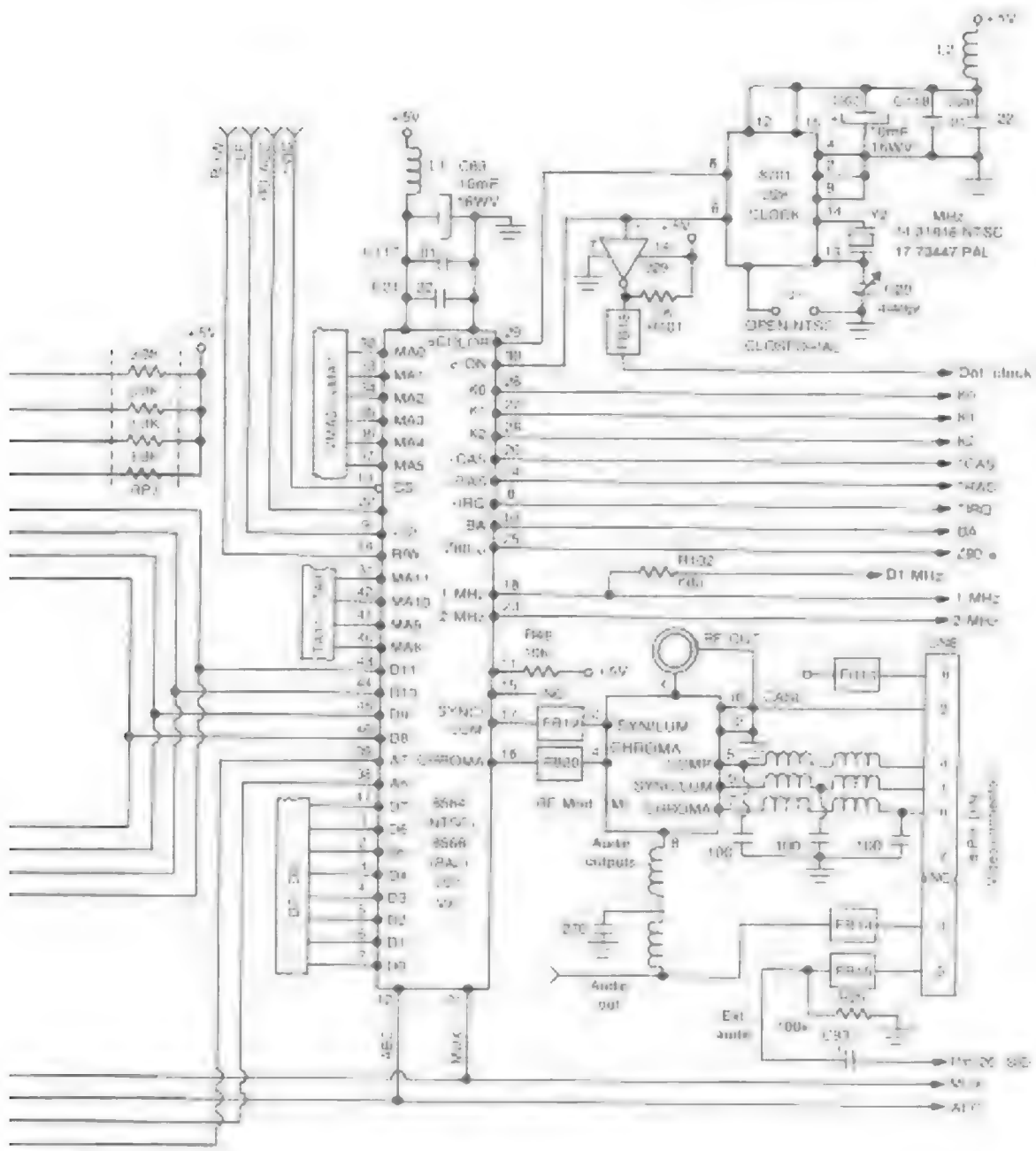


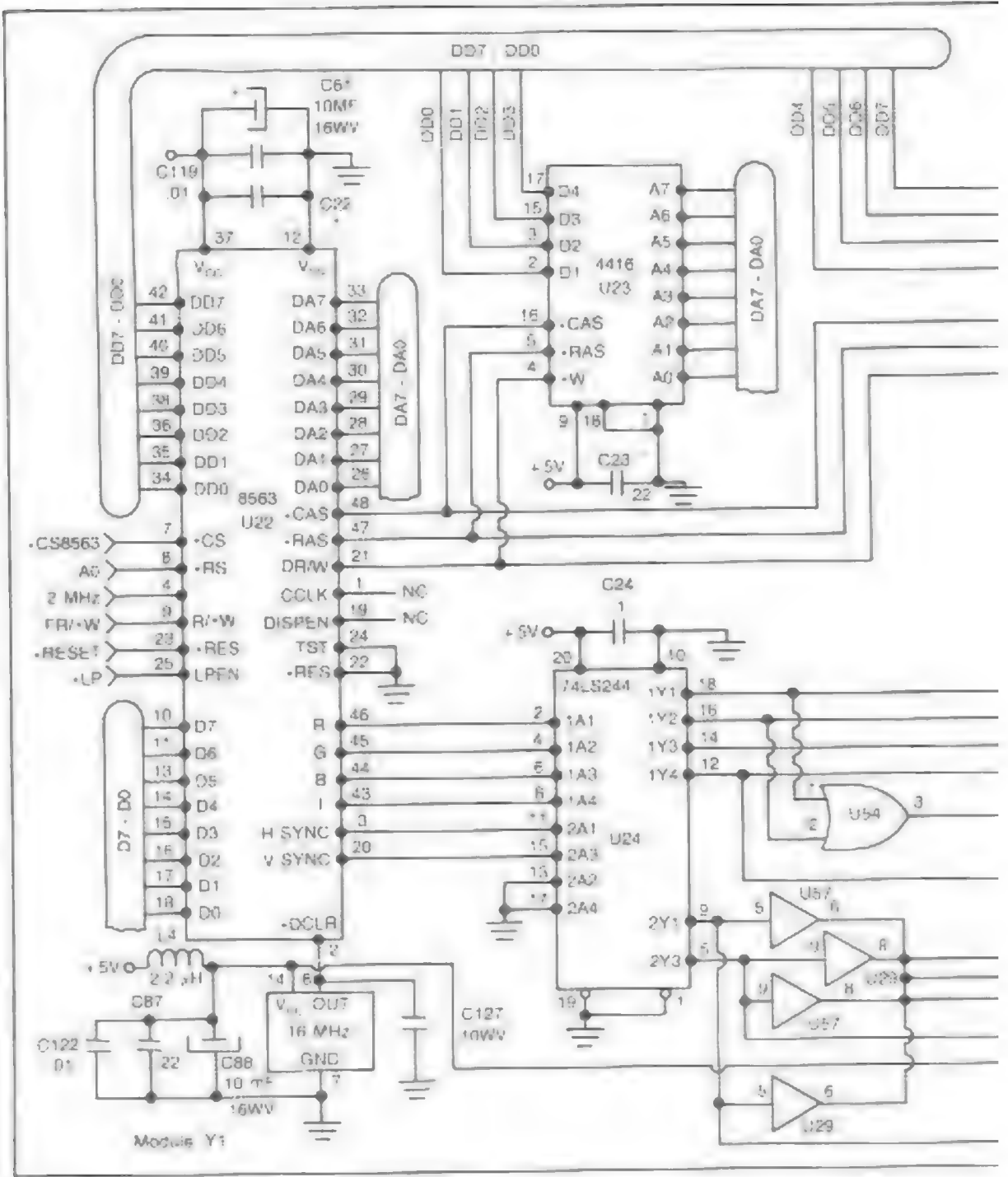


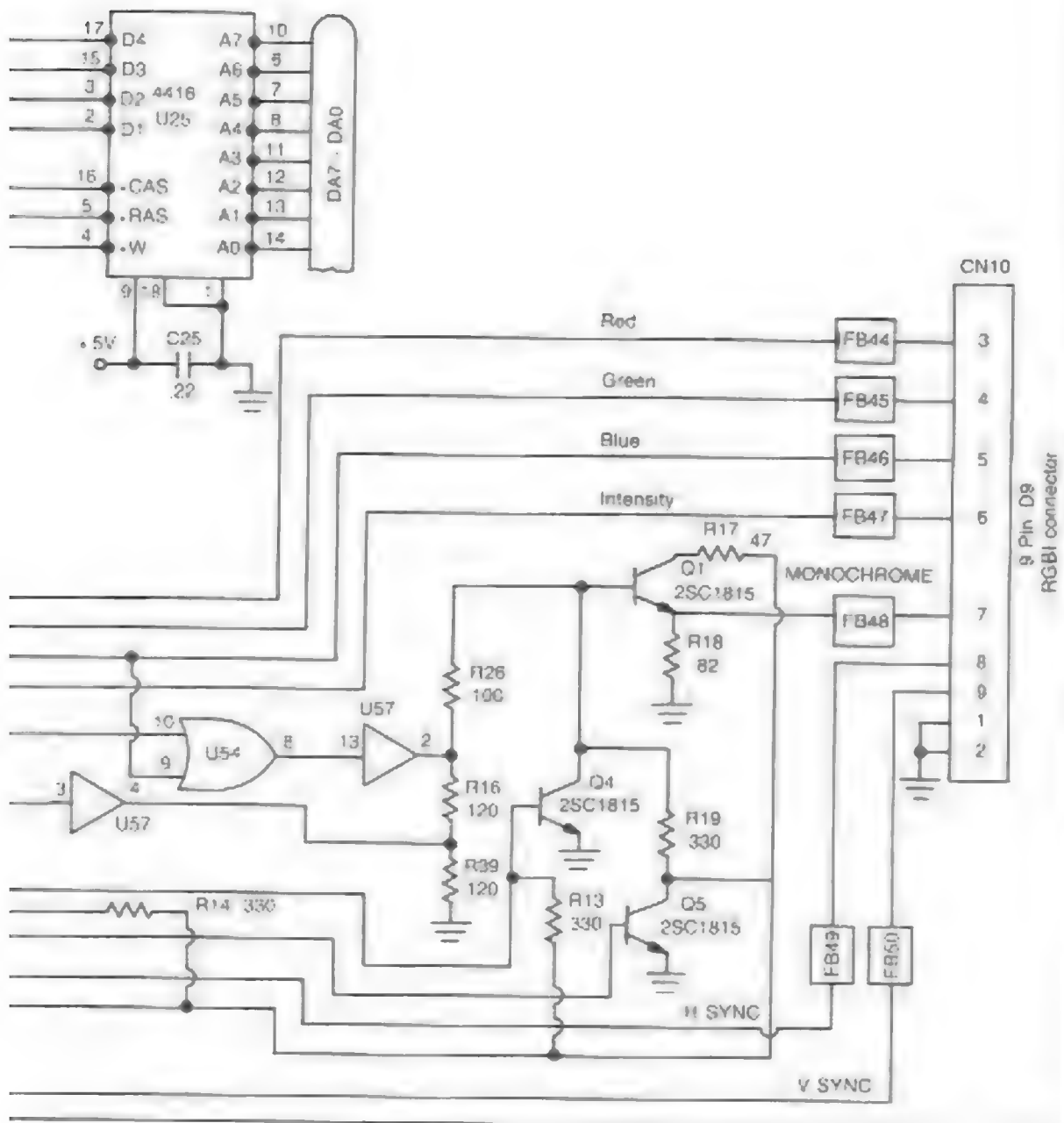


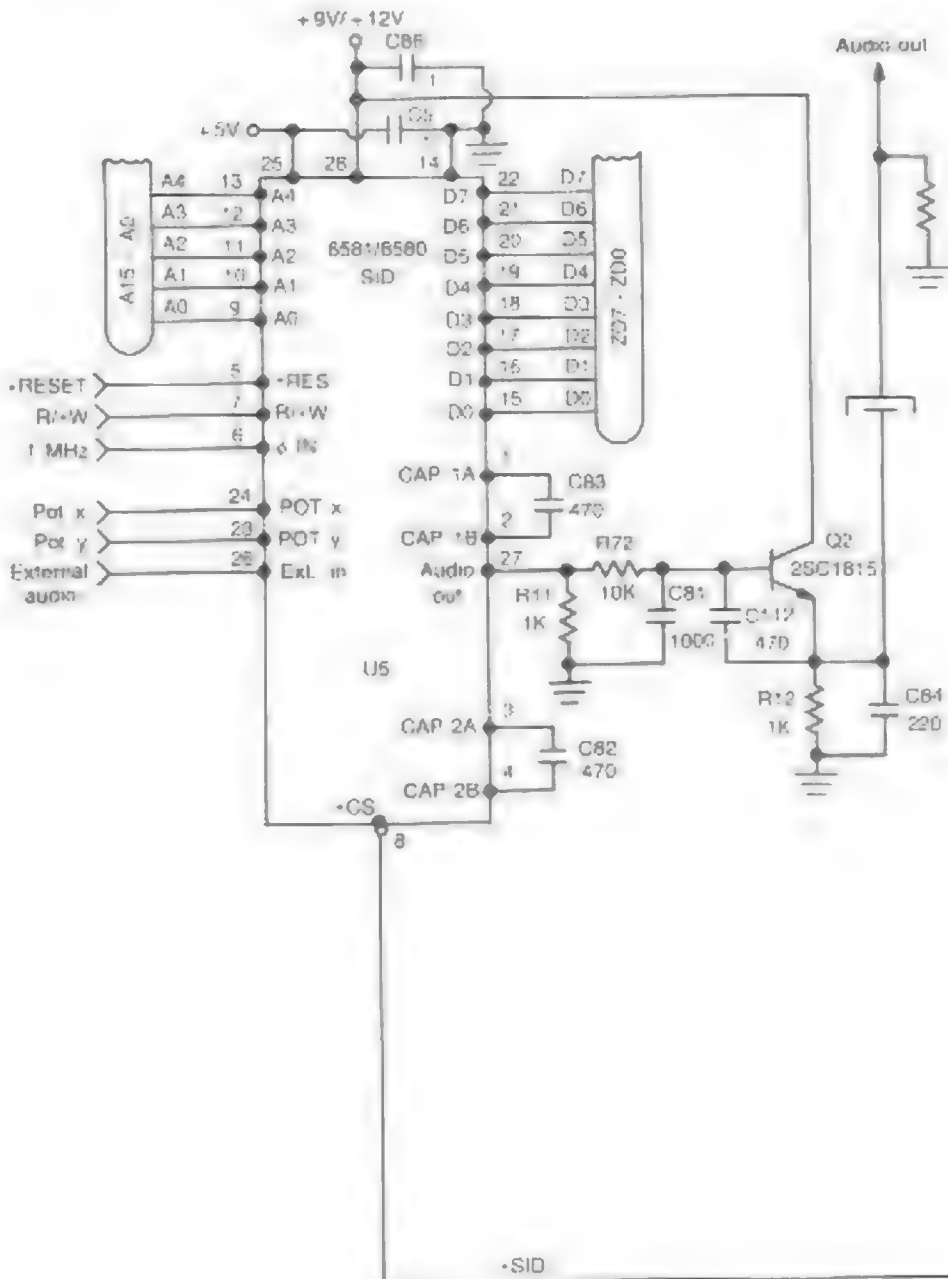


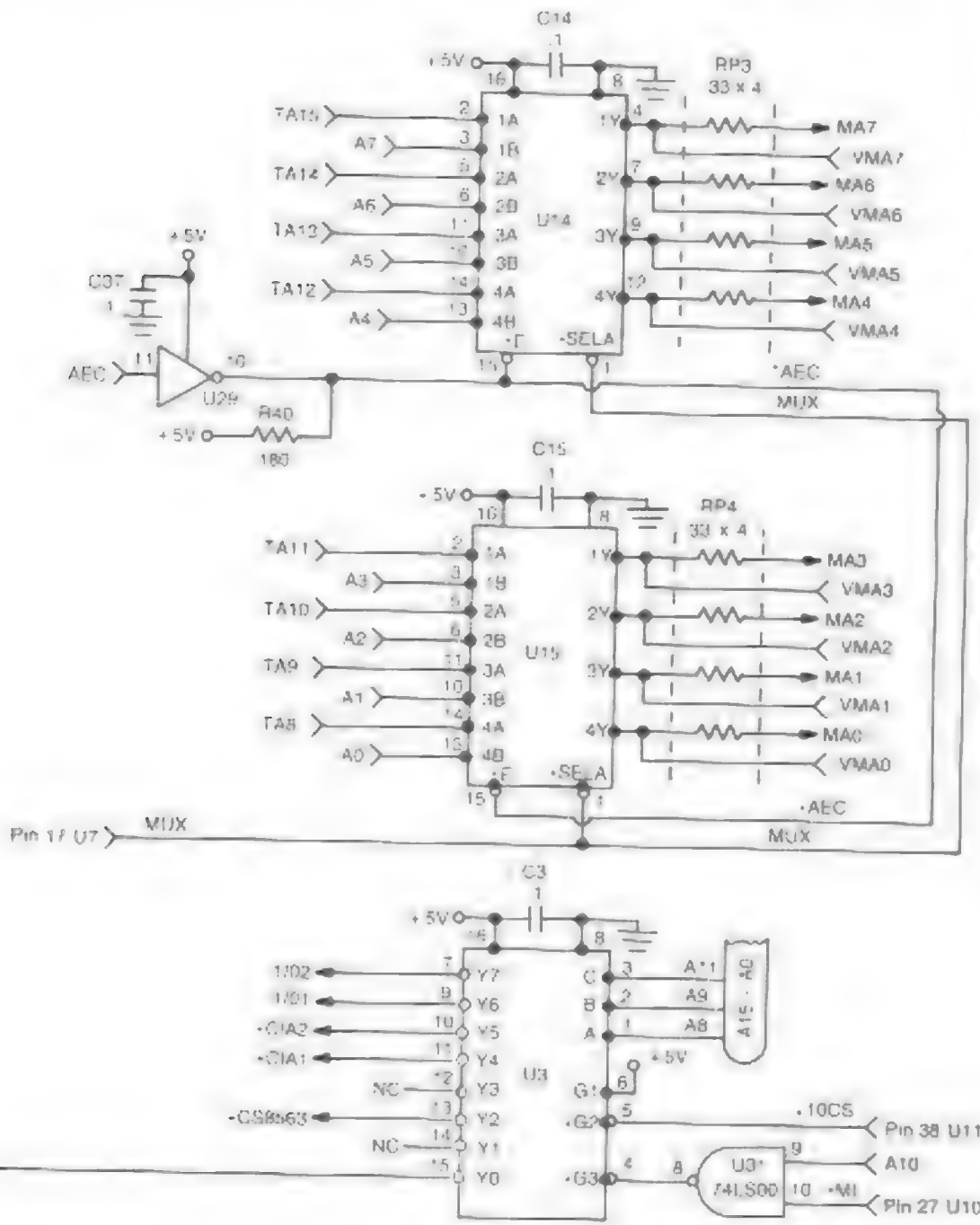












Index

A

- accumulator register, 850?
- microprocessor, 208, 211-213, 214
- address buses, 298-303
- address lines, 70, 71
- 8502 microprocessor, 204
- address register, 8563 video controller, 81
- address signals, 287
- addressing system, MPU location of, 15, 16
- AND gate, 132, 175, 179-182
 - circuit description of, 181
 - input and output for, 125
 - multiplexer use of, 136
 - nnp transistor in, 182
 - programmed logic array (PLA) and, 242
 - schematic and truth table for, 180
- ANDing, computing register, 202
- arithmetic logic unit (ALU), 43
- 8502 microprocessor, 209-211
 - clearing and complementing in, 210
 - flag register function with, 221
 - flagging in, 210
 - logic manipulations in, 210-211
 - rotating in, 211

shifting in, 210

system block diagram location of, 151

stable oscillator, 148

audio-video port, 39?

B

- BASIC
 - binary coding and, 167
 - ROM for, 104
- bilateral switch, quad, 4088, 143-146
- binary coding, 167
- binary counter, 192
- binary numbers, 169-170
- bipolar transistor, 50, 51, 56
- bit map modes, video interface chip (VIC), 348
- bits vs. decimal count, 89
- blackening of printboard, 29
- buffer, 138
- bus lines, 314-320
- byte size registers, 173

C

- C128 computer, memory maps for, 270-272
- C64 computer, memory maps location in, 272
- cassette I/O slot, 382

- character colors, video interface chip (VIC), 344
- character contents ROM, 108, checking out, 110
- character fetch, video interface chip (VIC), 342
- character modes, video interface chip (VIC) and, 345
- chip changing techniques, 49-66
- chip location guide, 39-48
 - system block diagram and, 150
- chip select pins
 - programmed logic array (PLA), 243
 - static RAM, 90
- chips
 - rugged, 49-53
 - sensitive, 55-58
 - U designation of, 41, 42
 - upside down, installation of, 41
- CIA1 and CIA2, 44, 335
- cleaning, 31-32
- clearing
 - ALU, 210
 - computing register, 197-198
- clocks, 280-297
 - 2MHz mode, 290
 - 8502 timing control and, 283-287
 - address signals and, 287

- control lines and, 314
- data timing and, 288
- other timing signals and, 267
- reading and writing to peripherals and, 288
- real time, complex interface adapter (CIA and, 323)
- size to square wave, 280-283
- test point chart for, 297
- testing of, 293-297
- VIC and, 281
- video dot, 8563 video controller and, 361
- Z80 and, 290-293
- CMOS chips, 55, 58
- CN1-CN10 ports and slots, 380-385
- color, displays, absence of, 7
- column address strobe, 97
- complementing
 - ALU, 210
 - computing register, 188-189
 - subtraction in, 199
- complex interface adapter (CIA), 10, 44, 72-77, 321-326
 - 74LS138 decoder and, 325
 - addressing and controlling, 78, 159, 321-323
 - handshaking operation in, 79
 - I/O ports for, 325
 - internal data transfer in, 323
 - interrupt control register in, 330
 - joystick interface function of, 74
 - keyboard interface function of, 73
 - location of, 13
 - MPI and, 158-160
 - PC and FLAG pins on, 326
 - ports on, 78
 - read/write line on, 160
 - real time clock for, 329
 - serial data register and, 330
 - serial I/O shift register in, 77
 - system block diagram location of, 151
 - testing of, 335
 - time of day (TOD) clock in, 77
 - timing in, 323
- composite video port, 8
- computing registers, 185-202
- connectors, Kern numbers for, 41
- control bus, test for, 319
- control lines, 310-314, 362
- control ports, 380
- counters, 183
- CPM mode, memory maps for, 274
- CR registers, memory management unit (MMU), 252

D

- data buses, 204, 302-308, 319
- data lines, 72

- data register, 8563 video controller, 61

- data release time, 326
- data timing, 293
- dead computer, 4-6
- decimal numbers, 168-170
- decoder, 131-134, 216
- decrement register, 199-202
- diagnostic programming, 1-13, 115-116
- digital register servicing, 189-203
- digital square wave, 146
- diode CR15, test point use of, 41
- disassembly, 20-33
- display bus, 308-310
- displays
 - CGA mode, 4
 - composite video port for, 8
 - dead computer, 4-6
 - devices for, 2
 - empty display block, 6
 - five signal types available for, 3
 - garbage, 6
 - no color, 7
 - no sound, 9
 - no video, sound okay, 7
 - normal, 2-4
 - RGBI port for, 8
 - snowy, 4
- drain, IGFET, 56
- driver, octal, 138
- DTL chips, 51, 52
- dual in-line packages (DIP), 56, 58-61
- LSI, 67-85

- dual linear, 558, 146
- dynamic random access memory (DRAM), 45, 81, 88-103
 - 8502 microprocessor and, 153-156
 - bit holders in, 85, 97
 - bit registers in, 88
 - choosing registers in, 155
 - data bus and address bus line functions in, 96
 - location of, 13, 45, 46, 87
 - memory layout in, 98-100
 - memory refresh in, 97
 - multiplex chips for addressing in, 100
 - operation of, 100
 - refresh timing in, 102-103
 - row address and column address strobe in, 87-98
 - system block diagram location of, 153
 - test point charts for, 358
 - timing in, 101

E

- empty display block, 6
- ENV 3 register, 375

- envelope generator circuit, 63, 371

F

- falling edge, 143
- fan input, 51
- fan test, 30, 37
- fetch and execute cycle, 71, 215, 217
- field effect transistor (FET), 56, 57
- filter capacitor, test point use of, 41
- FLAG pin, 326
- flag register, 210
 - 8502 microprocessor, 220-226
 - ALU function with, 221
 - interrupt and overflow, 225
 - programming and, 222-223
- flip-flops, 130, 188-195
 - 556 dual timer as, 184-195
 - 74LS373 D latch, 139, 191-194
 - 74LS74 D, 130-131, 190-191
 - pnp transistors in, 190
 - Q and D input, 143
 - RS, 182
- flowchart, trouble analysis 17-18
- fuses, 407

G

- garbage displays, 6
- gate, 56
- go/no-go continuity chart, 317
- graphics, video interface chip (VIC) and, 337

H

- handshaking, 79
 - complex interface adapter (CIA) and, 326
- hex buffer/drivers, 120
- hex inverter, 118-120, 122
- hexadecimal coding, 167
- hexadecimal numbers, 171, 172
- hood removal, 20-22
- hot chips, 30

I

- IGFET, 56
- increment register, 199-202
- index registers, 8502 microprocessor, 219
- inoperative computer, 237
- input and output, 379-387
 - checking troubles with, 380
 - CN1-CN10 ports and slots, 380-385
 - connectors for, 380
 - RF modulator for, 383
- input/output devices, diagnostic function of, 1
- instruction byte, 8502 microprocessor, 215

- instruction register, 8502
 - microprocessor, 204
 - instruction set
 - 8502 microprocessor, 213-219
 - Z80 microprocessor, 236
 - integrated circuits, 118-149
 - interrupt control register, 330
 - interrupt mask bit, 223
 - interrupt register, 223
 - video interface chip (VIC) and, 351
 - I/O control line, 310, 313
- J**
- jump register, 199-202
 - jump table, ROM, 114-115
- K**
- kernel ROM, 104, 112-114, 115
 - keyboard, 1, 26, 75, 385
- L**
- latches, 137
 - light pen, video interface chip (VIC)
 - and, 352
 - logic gate servicing, 167-187
 - logic probe, logic state testing with, 54, 55
 - logic states, 174
 - 8502 microprocessor, 229
 - test point chart for, 119
 - testing, 53
 - three-state, 177
 - variations in, 177
 - VOM and logic probe testing of, 54, 55, 120
 - Z80 microprocessor, 238
- long lead shorts, 28
 - low voltage continuity tester, bus line testing with, 315-318
- LSI chips, 57, 67-85
 - complex interface adapter (CIA) in, 72-77
 - microprocessor, 68
 - PLA and MMU as, 84-85
 - sound interface device (SID) as, 82
 - VIC and video controller chips, 77
- M**
- machine language monitor, 265
 - memory and fill with, 278-279
 - mail order sources, 53
 - MB3759 dc voltages, 405, 406
 - memory
 - DRAM layout, 98-100
 - K designation in, 86
 - memory management unit (MMU), 84-85, 249-265
 - bank contents in, 249-251
 - bank number selection in, 84
 - functions of, 261-263
 - input signal for, 263
 - inspecting registers in, 255-258
 - location of, 45
 - mode configuration register in, 256-258
 - page pointers in, 259-261
 - pinouts for, 261
 - RAM configuration register in, 256-259
 - registers for, 251-256
 - setting preconfiguration registers in, 254-255
 - setting the CR register in, 252-254
 - test point chart for, 260
 - translated address bus from, 264
 - version register for, 261
- memory map, 88, 244, 266-279
 - 8502 I/O port and, 268-270, 268 arrangement of, 268
 - C128 locations for, 270-272
 - C64 locations for, 272
 - CP/M mode, 274
 - memory and fill with, 278-279
 - PEEK and POKE and, 277
 - processors and modes to produce, 267
 - system block diagram location of, 161
 - memory matrix, 88
 - memory refresh, DRAM, 97
 - microprocessors
 - 16-lines of addressing system in, 15
 - address line function in, 70, 71
 - data line function in, 72
 - locations of, 16, 43
 - LSI, 68-72
 - system block diagram location of, 151, 152
 - mode configuration register, 255-258
 - mode/volume register, sound interface device (SID), 973
 - monitor (see displays)
 - monochrome composite TV signal, 139, 351
 - MOS chip, 55
 - static electricity in, 33
 - MPU
 - complex interface device and, 158-160
 - read only memory (ROM) and, 156-160
 - sound interface device and, 162-164
 - system block diagram location of, 158
 - video controller and, 164-166
 - video interface controller and, 162-162
 - multiplex chips, 100
 - multiplexed address bus, 300, 306
 - multiplexer, 74LS257, 134-137, 155
- N**
- NAND gate, 128, 132, 180
 - 74LS00 and 74LS03, 124
 - TTL, schematic of, 54
 - NMOS chips, 55, 57, 58
 - no color in display, 7
 - NOR gate, 186
 - NOT gate, 178-179
 - npn transistor
 - NOT gate using, 179
 - YES gate using, 178
 - number systems, 169-171
 - nybble, 88, 172, 193
- O**
- octal driver, 74LS244, 138
 - octal latches, 74LS373, 137-138
 - one-shot timer, 146
 - open collector, 127
 - operating system (OS), 104
 - OR gate, 128, 175, 183-185
 - circuit description of, 184
 - parallel diodes in, 184
 - programmed logic array (PLA) and, 242
 - quad input, 129
 - schematic and truth table for, 183
 - ORing, computing register, 202
 - OSC 3/random register, 375
 - oscillator, astable, 146
 - output voltage test locations, 41
 - overflow flag, 223
- P**
- page pointers, memory management unit (MMU), 259-261
 - parallel output, 77
 - PC pin, 326
 - PEEK, 10, 11, 67, 115
 - logic gate servicing and, 171
 - memory maps and, 277
 - register testing with, 191
 - peripherals
 - diagnostic function of, 1-10
 - reading and writing to, 288
 - pinch, 83
 - PMOS chips, 55, 57, 58
 - pnp transistor
 - AND gate using, 182
 - flip-flop using, 190
 - POKE, 10, 67, 115
 - logic gate servicing and, 171
 - memory maps and, 277
 - register testing with, 191
 - sound interface device (SID) testing with, 977
 - popping, 209
 - ports, chip designation (CN for, 39

pot register, 374
power regulator, 400, 403
power supply, 5, 398-405
power supply box, 35
 disassembly of, 34
 removal of, 24-27
power transformer, 399, 403
preconfiguration registers, memory management unit (MMU), 254
printboard
 blackening on, 29
 chip location guide for, 38-48
 cleaning of, 31
 long lead shorts in, 28
 position of and insulators for, 36
 soldering defects on, 27
 visual defects in, 29
printboard landmarks, 13-17, 43-48
printboard removal, 22-23
processor access, 305
program counter, 200
programmable logic array (PLA), 45, 84-85, 240-248
 additional functions of, 243-245
 AND and OR gates in, 242
 chip selects for, 243
 internal functions of, 240-245
 memory map selection by, 244, 245
 pinouts for, 246-248
 schematic diagram of, 241
 system block diagram location of, 151

Q

quad bilateral switch, 143-148
quad input AND gates, 121-124
quad input OR gate, 129

R

RAW control line, 310, 311
RAW hold and setup time, 325
RAM configuration register, memory management unit (MMU), 256-259
random access memory (RAM), 87
raster register, 350
RD and WR control lines, 310, 312
read only memory (ROM), 61, 104-117
 block diagram of, 107-110
 burning in programs in, 104-108
 contents of, 108, 110, 111-112
 diagnostic programs for, 115-116
 internal operations in, 47, 109
 jump tables for, 114-115
 kernel, 112-114
 kernel, monitor, and BASIC, 110-111
 location of, 13, 46, 47
 logic probe test for, 112

MPU and, 156-158
 row location in, 106
 short routine for, 116-117
 system block diagram location of, 151, 156
 test point chart for, 108
 timing signals for, 113
read timing, complex interface adapter (CIA), 325
read/write lines, 8563 video controller and, 362
read/write setup time, 287
reading/writing to peripherals, 288
real-time clock, 329
refresh timing, DRAM, 102-103
registers

 8563 video controller, 356
 digital, servicing of, 188-203
 logic gates, 188
 memory management unit (MMU), 251-256
 non-programmable, Z80 microprocessor, 235
 PEEK and POKE to test, 11, 191
 sound interface device (SID), 369
 static RAM, 68
 update RAM location, 359
 Z80 microprocessor 232-235
RESET control line, 313, 315
RF modulator box, 31, 39
 circuitry of, 163
 input and output and, 389
 location of, 46
 outputs of, 164
 system block diagram location of, 162

RGB? port, 9, 395
rising edge, 149
rotating, ALU, 211
row address strobe, 97
RS flip-flop, 192
RTL chips, 51, 52
rugged chips, 49-53

S

schematics, 408-429
Schmitt trigger, 74LS14, 141, 143
Schottky diode, 63, 121-124
scratch pad (see accumulator register)
sensitive chips, 55-58
serial data register, 330
serial I/O shift register, 77
serial output, 77
serial ports, 368
shared address bus, 302, 305
shift register, 196-197
shifting
 ALU, 210
 computing register, 196-197
shorts, long lead, 28

sine wave, 280-282
snowy display, 4
socketed chips, 61-64
soldered-in chips, 64-66
soldering, 64, 68
 defects in, 27, 36
 sound, 7, 8-10
sound effects generator, 83
sound interface device (SID), 9, 17, 62-64, 364-378
 emulator generator registers, 371, 375
 filter registers, 373
 location of, 45
 mode/volume register in, 373
 MPU and, 162-165
 operation of, 367-369
 OSC 3/random register, 375
 other voices in, 373
 pinout for, 364-367
 pot registers in, 374
 sound effects generator function of, 83
 special inputs in, 367
 system block diagram location of, 153
 testing of, 377
 timing in, 377
 voice control register in, 371
 writing and reading registers in, 374
sprites, video interface chip (VIC), 347-349
square wave, sine wave to, 280
stack, 148
 8502 microprocessor, 204-209
 pointer, 206, 207
static electricity hazard, 32-38
static RAM, 67-92
 internal chip organization for, 84
substrate, 4GFET, 56
subtraction by addition of complements, 199
switches, 39, 41
synonym devices, 1-10
system block diagram, 150-188
 complex interface adapter location in, 151
 DRAM location in, 153
 memory map location in, 151
 microprocessor location on, 151, 152
 MPU and CIA interface at, 158-160
 MPU and ROM in, 160
 MPU and sound interface device in, 162-164
 MPU and video controller in, 164-166
 MPU and video interface controller in, 160-162

programmable logic array (PLA)
location in, 151
RF modulator box location in, 162
ROM chips in, 151
sound interface device location in,
153
video controller location in, 153
video interface controller (VIC) lo-
cation in, 151

T

test point charts, 119
4066 quad bilateral switch, 148
4418 DRAM, 358
7406 hex inverter, 122
7407 hex buffer/drivers, 123
74F245 transceiver, 144
74F32 quad input OR gate, 130
74LS00 and 74LS03 NAND gate,
126
74LS08 quad input AND gate, 124
74LS14 Schmitt trigger, 145
74LS244 octal driver, 143
74LS257 multiplexer, 138
74LS32 quad input OR gate, 130
74LS373 octal latch, 141
8563 video controller, 360
8701 clock, 297
test programs, writing of, 12
three-state, 177
three-state buffers, 8502
microprocessor, 205
threshold voltage for electrostatic
damage, 33
time of day (TOD) clock, 77
timers, 148
complex interface adapter (CIA)
and, 326
registers for, 328
timing
control, 283-287
DRAM, 101
signals, 287
timing, 65
transceiver, 74F245, 141
transistors, 39, 41, 48, 50, 56
translated address bus, 300
installing TTL chips, 53-55
trouble analysis, 2, 17-19

troubleshooting charts and
schematics, 406-429
truth tables, 175
TTL chips, 32, 50, 53-55, 61

U

U number chips, parts number and
given names for, 42
U28 16-pin DIP, 148, 280
U59 IC, 148
update RAM location registers, 359
user port, 394

V

version register, 261
video controller, 44, 61, 78, 88, 309,
354-363
address and data registers in, 81
attributes for, 358
character block color for, 358
control lines for, 362
monochrome composite TV signal
from, 361
MPU and, 164-166
pin-by-pin checkout for, 359-363
pinout diagram for, 80, 82
RAM for, 355-356
read/write lines in, 362
registers for, 358
system block diagram of, 153
test point chart for, 360
video dot clock, 361
video display, sound okay, video ab-
sent in, 7
video dot clock, 8563 video con-
troller and, 361
video interface chip (VIC), 17, 44, 61,
77, 337-354
addressing RAM for, 304
bit map modes, 346
character colors, 344
character fetch, 342
character modes, 345
clock origination in, 281
functions of, 161
graphics for, 337
inside clock operation in, 282-283
interrupt register in, 351
light pen with, 352

MPU and, 160-162
pin-by-pin operation of, 338,
339-347
rasler register in, 350
screen blanking with, 349
sprites and, 347-349
system block diagram location of,
151
testing of, 353
video output and, 352
visual repairs, 27-30
voices, 83
voltage comparator circuit, 184
VOM, 120
logic state testing with, 54, 55

W

word registers, 173
wrist strapping, 37
write timing, complex interface
adapter (CIA), 323

X

XNOR gate, 186
XOR gate, 185

Y

YES gate, 173-178

Z

Z80 microprocessor, 43, 61,
231-239
address bus origination in, 288
block diagram of, 231-235
clocks for, 290-293
connections for, 236
inoperative computer and, 203,
237
instruction set for, 236
logic states for, 238
non-programmable registers for,
235
registers in, 232-235
schematic diagram of, 235-236
system address bus connection
for, 234
system block diagram location of,
151

Troubleshooting and Repairing Your **COMMODORE 128**

ART MARGOLIS

**Hundreds of do-it-yourself repairs to save you
time, money, and computer down time!**

This sourcebook is probably the most important "add-on" you'll ever buy for your C-128! Packed with maintenance tips, troubleshooting procedures, and do-it-yourself repair techniques, it can save you time, money, and untold frustration over the lifetime of your Commodore-128.

The simple chip-changing instructions alone will enable you to cure over 50% of the problems that cause breakdowns. Plus, Margolis gives you all the technical detail and advanced computer repair guidance that you'll need! Both advanced hardware enthusiasts and professional service technicians will appreciate the in-depth examination of every C-128 circuit—including timing diagrams, and the use of PEEK and POKE commands to signal-trace circuits.

If you're a programmer interested in getting maximum performance from your machine, then you, too, need look no further than this book. The information provided here will help you gain mastery of your computer. In addition to complete memory maps and thorough descriptions of the C-128's operating characteristics, you'll find:

- Tried and true service procedures for pinpointing trouble spots.
- A Chip Location Guide—a layout diagram of all 63 chips.
- Electronic and mechanical techniques for chip replacement.
- Test Point Charts showing exact pinouts, the name of each pin, arrows indicating direction of signal flow, and readings you should receive using a logic probe or vom.
- A master schematic of the Commodore 128, complete with all part numbers—an essential tool for more difficult repairs.

A better understanding of how your C-128 works will help make you a better, more informed programmer. Also, with the high cost of professional computer repairs, just being able to replace one chip will more than pay for this valuable guide!

Art Margolis is a computer professional who has written several best-selling books including *Computer Technician's Handbook—2nd Edition*, *Troubleshooting and Repairing Personal Computers*, and *Troubleshooting and Repairing Your Commodore 64™*.

TAB

TAB BOOKS Inc.

Blue Ridge Summit, PA 17294-0850

\$18.95

ISBN 0-8306-9399-8

PRICES HIGHER IN CANADA

1088